Raouf Boutaba Kevin Almeroth
Ramon Puigjaner Sherman Shen
James P. Black (Eds.)

LNCS 3462

NETWORKING 2005

Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems

4th International IFIP-TC6 Networking Conference
Waterloo, Canada, May 2005
Proceedings

ifip

Networking 2005

Springer

Lecture Notes in Computer Science 3462

Raouf Boutaba   Kevin Almeroth
Ramon Puigjaner   Sherman Shen
James P. Black (Eds.)

NETWORKING 2005

Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication Systems

4th International IFIP-TC6 Networking Conference
Waterloo, Canada, May 2-6, 2005
Proceedings

Springer

Volume Editors

Raouf Boutaba
Sherman Shen
James P. Black
University of Waterloo
School of Computer Science
200 University Ave. West, Waterloo, Ontario, N2L 3G1, Canada
E-mail: {rboutaba/sshen/jpblack}@uwaterloo.ca

Kevin Almeroth
University of California
Department of Computer Science
Santa Barbara, CA 93106-5110, USA
E-mail: almeroth@cs.ucsb.edu

Ramon Puigjaner
Universitat de les Illes Balears
Dept. de Ciències Matemàtiques i Informàtica
Cra. de Valldemossa, km. 7.5, 07122 Palma, Illes Balears, Spain
E-mail: putxi@uib.es

Library of Congress Control Number: Applied for

CR Subject Classification (1998): C.2, C.4, H.4, D.2, J.2, J.1, K.6, K.4

ISSN 0302-9743
ISBN-10 3-540-25809-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25809-4 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11422778 06/3142 5 4 3 2 1 0

Preface

Welcome to the proceedings of Networking 2005, the fourth conference in what is now an annual series. This was the first time the conference was held outside Europe, and the University of Waterloo is proud to be the institutional host.

Networking is the flagship conference of Technical Committee 6, Communication Systems, of the International Federation for Information Processing (IFIP TC 6). The conference is sponsored by four of the eleven working groups in TC 6: Network and Internetwork Architectures (WG 6.2), Performance of Communication Systems (WG 6.3), Wireless Communications (WG 6.8), and Photonic Networking (WG 6.10).

Previously unpublished technical papers were solicited and peer-reviewed in three tracks: Networking Technologies, Services and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communication Systems. The final program reflected this organization as much as possible. In addition, four proposals for high-quality workshops were accepted, there were three keynote speeches and three panel sessions, and a program of tutorials was put in place. The Organizing Committee endeavored to complement the exciting technical program with an attractive social program.

The Technical Program Committee, lead by Raouf Boutaba, accepted 105 full papers and 36 poster papers from 430 submissions, originating in 26 countries on four continents. This placed a large load on the TPC and track chairs, the 199 members of the TPC, and more than 600 reviewers who wrote over 1800 reviews.

An international conference of this scope only occurs when many, many individuals contribute in many ways to its success. We wish to thank the University of Waterloo for the use of its facilities, conference services, and food services. We are grateful to the Nortel Networks Institute at the University of Waterloo for financial support of the publication of the proceedings by Springer. The Centre for International Governance Innovation graciously allowed the use of their stunningly beautiful atrium for the conference banquet. Members of the Organizing Committee worked for many months to make the conference a reality, and the Steering Committee was always present for advice and information. As the main sponsor, IFIP provided the impetus for the conference, as well as an award for the best paper and a number of student travel grants.

Finally, and most importantly, we thank the many reviewers and members of the Technical Program Committee for ensuring the academic vitality of Networking 2005.

May 2005

Raouf Boutaba
James P. Black

Organization

Networking 2005 was organized by the University of Waterloo, Canada, and sponsored by the IFIP working groups on Network and Internetwork Architectures (WG 6.2), Performance of Communication Systems (WG 6.3), Wireless Communications (WG 6.8), and Photonic Networking (WG 6.10).

Executive Committee

General Chair:	Jay Black (University of Waterloo, Canada)
Program Chair:	Raouf Boutaba (University of Waterloo, Canada)
Chair, Networking Technologies, Services and Protocols Track:	Kevin Almeroth (University of California, Santa Barbara, USA)
Chair, Performance of Computer and Communication Networks Track:	Ramon Puigjaner (Universitat de les Illes Balear, Spain)
Chair, Mobile and Wireless Communications Track:	Sherman Shen (University of Waterloo, Canada)
Keynote and Panel Co-chairs:	Edmundo Monteiro (University of Coimbra, Portugal)
	Catherine Rosenberg (University of Waterloo, Canada)
Tutorial and Workshop Co-chairs:	Lorne Mason (McGill University, Canada)
	Alex Lopez-Ortiz (University of Waterloo, Canada)
Publicity Co-chairs:	Guy Omidyar (Sultan Qaboos University, Oman)
	Paul Ward (University of Waterloo, Canada)
Publication Chair:	Thomas Kunz (Carleton University, Canada)
Industrial Relations Chair:	Vic Diciccio (University of Waterloo, Canada)
Local Arrangements:	Wendy Rush (University of Waterloo, Canada)
	Jean Webster (University of Waterloo, Canada)
Webmaster:	Herman Li (University of Waterloo, Canada)
Steering Committee:	Harry Perros (Chair, North Carolina State University, USA)
	Augusto Casaca (IST/INESC, Portugal)
	Guy Omidyar (Sultan Qaboos University, Oman)
	Guy Pujolle (University of Paris 6, France)
	Otto Spaniol (Aachen University, Germany)
	Ioannis Stavrakakis (University of Athens, Greece)

Program Committee

Program Chair: Raouf Boutaba, University of Waterloo, Canada

Networking Technologies, Services and Protocols
Chair: Kevin Almeroth, University of California, Santa Barbara, USA
Mostafa Ammar, Georgia Tech, USA
Samrat Bhattacharjee, University of Maryland, USA
Andrea Bianco, Politecnico di Torino, Italy
Milind Buddhikot, Bell Labs, USA
Claudio Casetti, Politecnico di Torino, Italy
Jun-Hong Cui, University of Connecticut, USA
Jordi Domingo-Pascual, University of Catalunya, Spain
Sonia Fahmy, Purdue University, USA
Wu-chi Feng, Oregon Graduate Institute, USA
Maurice Gagnaire, ENST, France
Manimaran Govindarasu, Iowa State University, USA
David Hutchison, Lancaster University, UK
Gianluca Iannaccone, Intel Research Cambridge, USA
Shivkumar Kalyanaram, Rensselaer Polytechnic Institute, USA
Kimon Kontovasilis, Demokritos Research Center, Greece
Guy Leduc, University of Liege, Belgium
Jorg Liebeherr, University of Virginia, USA
Peter Marbach, University of Toronto, Canada
Lorne Mason, McGill University, Canada
Ketan Mayer-Patel, University of North Carolina, USA
Michael Menth, University of Wuerzburg, Germany
Vishal Misra, Columbia University, USA
Prasant Mohapatra, University of California, Davis, USA
Ioanis Nikolaidis, University of Alberta, USA
Peng Ning, North Carolina State University, USA
Jaudelice de Oliveira, Drexel University, USA
Christos Papadopoulos, University of Southern California, USA
Symeon Papavassiliou, New Jersey Institute of Technology, USA
Erwin Rathgeb, University of Essen, Germany
Reza Rejaie, University of Oregon, USA
George Rouskas, North Carolina State University, USA
Saswati Sarkar, University of Pennsylvania, USA
Stefan Saroiu, University of Toronto, Canada
Ashwin Sridharan, Sprint ATL, USA
Aaron Striegel, University of Notre Dame, USA
Lars Wolf, Braunschweig University of Technology, Germany
Richard Yang, Yale University, USA
Daniel Zappala, University of Oregon, USA
Ben Zhao, University of California, Santa Barbara, USA

Performance of Computer and Communication Networks
Chair: Ramon Puigjaner, Universitat de les Illes Balear, Spain
Ron Addie, University of Southern Queensland, Australia
Eitan Altman, INRIA, France
Miltiades Anagnostou, National Technical University of Athens, Greece
Andrea Baiocchi, University “La Sapienza”, Rome, Italy
Chris Blondia, University of Antwerp, Belgium
Tosten Braun, University of Bern, Switzerland
Herwig Bruneel, University of Ghent, Belgium
Mariacarla Calzarossa, University of Pavia, Italy
Constantinos Courcoubetis, Athens University of Economics and Business, Greece
Khaled Elsayed, Cairo University, Egypt
Sebastià Galmés, Universitat de les Illes Balears, Spain
Jorge García, Universitat Politècnica de Catalunya, Spain
Guenter Haring, University of Vienna, Austria
Peter Harrison, Imperial College, London, UK
Ilias Iliadis, IBM Research, Switzerland
Farouk Kamoun, Université de la Manouba, Tunisia
Krishna Kant, Intel Corporation, USA
Peter Key, Microsoft Research Ltd., Cambridge, UK
William Knottenbelt, Imperial College, London, UK
Demetres Kouvatsos, University of Bradford, UK
Pieter Kritzinger, University of Cape Town, South Africa
Emilio Leonardi, Politecnico di Torino, Italy
Alberto Leon-Garcia, University of Toronto, Canada
Michela Meo, Politecnico di Torino, Italy
Edmundo Monteiro, University of Coimbra, Portugal
Ilkka Norros, VTT Technical Research Centre of Finland, Finland
Luis Orozco, Universidad de Castilla-La Mancha, Spain
Harry Perros, North Carolina State University, USA
Andreas Pitsillides, University of Cyprus, Cyprus
Ana Pont, Universitat Politècnica de València, Spain
Otto Spaniol, Rheinisch-Westfälische Technische Hochschule Aachen, Germany
Yutaka Takahashi, University of Kyoto, Japan
Yannis Viniotis, North Carolina State University, USA

Mobile and Wireless Communications
Chair: Sherman Shen, University of Waterloo, Canada
Ozgur Akan, Middle Eastern Technical University, Turkey
Eitan Altman, INRIA, France
Luciano Bononi, University of Bologna, Italy
Hsiao-Hwa Chen, National Sun Yat-sen University, Taiwan
Marco Conti, National Research Council, Italy
Christina Czernuszka, Telus, Canada
Eylem Ekici, Ohio State University, USA

Laura Feeney, Swedish Institute of Computer Science, Sweden
Gabor Fodor, Ericsson, Sweden
Jerome Galtier, France Télécom R&D and INRIA, France
Silvia Giordano, ICA-DSC-SUPSI, Switzerland
Enrico Gregori, National Research Council, Italy
Ekram Hossain, University of Manitoba, Canada
Youssef Iraqi, University of Waterloo, Canada
Robin Kravets, University of Illinois at Urbana-Champaign, USA
Thomas Kunz, Carleton University, Canada
Victor Leung, University of British Columbia, Canada
Chuang Lin, Tsinghua University, China
Gerald Maguire, Royal Institute of Technology, Sweden
Jelena Mišić, University of Manitoba, Canada
Zhisheng Niu, Tsinghua University, China
Stephan Olariu, Old Dominion University, USA
Sergio Palazzo, University of Catania, Italy
Jianping Pan, NTT, USA
Nikos Passas, University of Athens, Greece
George Polyzos, Athens University of Economics and Business, Greece
Guy Pujolle, Université Pierre et Marie Curie (Paris 6), France
Kimmo Raatikainen, University of Helsinki, Finland
Aladdin O.M. Saleh, Bell Mobility, Canada
Ivan Stojmenovic, University of Ottawa, Canada
Violet Syrotiuk, Arizona State University, USA
Shahrokh Valaee, University of Toronto, Canada
Andras Valko, Ericsson, Sweden
Bernhard Walke, Aachen University, Germany
Lichun Wang, National Chiao Tung University, Taiwan
Gualiang (Larry) Xue, Arizona State University, USA
Richard Yao, Microsoft Research Asia, China
Xiaohu You, Southeast University, China
Ping Zhang, Beijing University of Post and Telecommunications, China
Qian Zhang, Microsoft Research Asia, China
Yongbing Zhang, University of Tsukuba, Japan
Dongmei Zhao, McMaster University, Canada

Reviewers

Atef Abdrabou
Anthony Acampora
Naotoshi Adachi
Ronald Addie
Ishfaq Ahmad
Ozgur Akan
Ozdemir Akin
Ian Akyildiz
Roberto Albanese
Guido Albertengo
Kevin Almeroth
Eitan Altman
Mostafa Ammar
Miltos Anagnostou
Akkihebbal L. Ananda
Emilio Ancillotti
Annamalai Annamalai
Josephina Antoniou
John Apostolopoulos
Supavadee Aramvith
Ashok Argent-Katwala
Jari Arkko
Susanna Au-Yeung
Irfan Awan
Amr Ayad
Vahid Azhari
Sang Bae
Vittorio Bagini
Pradeep Bahl
Andrea Baiocchi
S. Balasubramaniam
Ilia Baldine
Ashok Baldotra
Ashok Kumar Baldotra
Nilanjan Banerjee
Luis Barbosa
Jose Barcelo
Antonio Barili
Peter Barta
Lina Battestilli
Roberto Battiti
Daniel Bauer
Florian Baumgartner
Christian Becker
Abdelfettah Belghith
Omar Benali
Peter Benko
Christian Bettstetter
Konstantin Beznosov
Samrat Bhattacharjee
Amiya Bhattacharya
Supratik Bhattacharyya
Mauro Biagi
Andrea Bianco
Andreas Binzenhoefer
Danilo Blasi
Chris Blondia
Thomas Bohnert
Luciano Bononi
Eleonora Borgia
Nizar Bouabdallah
Ali Boudani
Raouf Boutaba
Arnold Bragg
Torsten Braun
Marc Brogle
Herwig Bruneel
Raffaele Bruno
Milind Buddhikot
Nirupama Bulusu
Ioana Burcea
Levente Buttyan
Alberto Cabellos-Aparicio
Filippo Cacace
Mario Cagalj
Jun Cai
Maria Calzarossa
Andrew Campbell
Srdjan Capkun
Licia Capra
Claudio Casetti
Llorenç Cerdá
Krishnendu Chakrabarty
Y. Chandramouli
Girish Chandranmenon
Dongyan Chen
Hsiao-Hwa Chen
Qing Chen
Wen-Tsuen Chen
Xiaobo Chen
Yue Chen
Zhen Chen
Yu Cheng
Roman Chertov
Jan Cheyns
Carla-Fabiana Chiasserini
Imrich Chlamtac
Sunghyun Choi
Christophoros Christophorou
Chrysostomos Chrysostomou
Andrea Clementi
Marco Conti
Costas Courcoubetis
Razvan Cristescu
Jun-Hong Cui
Laurie Cuthbert
Christina Czernuszka
Hui Dai
Sajal Das
Samir Das
Subir Das
Partha Dasgupta
Tugrul Dayar
Juan Carlos De Martin
Jaudelice de Oliveira
Andrea De Vendictis
Danny De Vleeschauwer
Franca Delmastro
Jing Deng
Ajay Dholakia
Nicholas Dingle
Jordi Domingo-Pascual
André Drummond
David Du
Rudra Dutta
Eylem Ekici
Brandon Ellison
Khaled Elsayed

Vincenzo Eramo
Fazli Erbas
Do Young Eun
Farid Fadaie
Sonia Fahmy
Afshin Fallahi
Yuguang Fang
Ramy Farha
Babak Fariabi
Jabed Faruque
Laura Feeney
Benjamin Feng
Kai-Ten Feng
Wu-chi Feng
David Filani
Joe Finney
Jorge Finochietto
Marco Fiore
Gabor Fodor
Viktoria Fodor
Mirko Franceschinis
Mario Freire
Rod Fretwell
Anders Furuskar
Maurice Gagnaire
Giulio Galante
Sebastia Galmes
Jerome Galtier
Shashidhar Gandham
Deepak Ganesan
Ayalvadi Ganesh
Wilfried Gansterer
Alberto García Martínez
María Luisa García-Osma
Jorge Garcia
Vangelis Gazis
Dipak Ghosal
Monisha Ghosh
Paolo Giaccone
José Gil
Silvia Giordano
Enrico Maria Giraudo
Bernd Girod
Tom Goff
José Luis González-Sánchez
Manimaran Govindarasu
Bob Gray
Enrico Gregori
Adelbert Groebbens
Dirk Grunwald
Chao Gui
Vehbi Cagri Gungor
Meng Guo
Ozgur Gurbuz
Andrei Gurtov
Rick Ha
Robert Haas
Reda Haddad
George Hadjipollas
Richard Han
Uli Harder
Khaled Harfoush
Guenter Haring
Peter Harrison
Masoud Hashemi
Seyed Naser Hashemi
Qi He
Yuning He
Wendi Heinzelman
Marc Heissenbuettel
Abdelsalam Helal
Ahmed Helmy
Jeffrey Hightower
Abdulrahman Hijazi
Helmut Hlavacs
Pin-Han Ho
Ekram Hossain
Xiaomeng Huang
Yueh-Ming Huang
Danny Hughes
Tian Hui Hui
Karin Hummel
Paul Hurley
David Hutchison
Gianluca Iannaccone
Ilias Iliadis
Mary Ingram
Youssef Iraqi
Brent Ishibashi
Mohammad Islam
Teerawat Issariyakul
Robert Istepanian
Eisaburo Itakura
Ravishankar Iyer
J.H. James
Ralf Jennen
Hai Jiang
Yixin Jiang
Emil Jovanov
Jiang Jun
Shivkumar Kalyanaram
Farouk Kamoun
Intae Kang
Krishna Kant
Martin Kappes
Dilek Karabudak
Naoyuki Karasawa
Ahmed Karmouch
Shoji Kasahara
Yoshihiro Kawahara
George Kesidis
Csaba Keszei
Peter Key
Babak H. Khalaj
Wolfgang Kiess
Jorma Kilpi
Myung Sup Kim
Seong-Lyun Kim
Aim Kind
Naoki Kita
William Knottenbelt
Masayoshi Kobayashi
Kimon Kontovasilis
David Kotz
Dimitrios Koutsouris
Demetres Kouvatsos
Andrey Krendzel
Pieter Kritzinger
Adlen Ksentini
Ferenc Kubinszky
Fernando Kuipers
Thomas Kunz
Ioannis Lambadaris

George Lampropoulos
Koen Langendoen
Nikolaos Laoutaris
Eddie Law
Long Le
Guy Leduc
Ming Lei
John Leis
Ilhem Lengliz
Alberto Leon-Garcia
Emilio Leonardi
Victor Leung
Bo Li
Chengzhou Li
Jiandong Li
Lei Li
Yue Li
Zheng Li
Zhi Li
Zhi Li
Ben Liang
Wanjiun Liao
Jorg Liebeherr
Noura Limam
Chuang Lin
Tao Lin
Tsung-Nan Lin
Xiaojun Lin
Xinhua Ling
Donggang Liu
Jing Liu
XiaoShan Liu
Yanbin (Grace) Liu
Renato Lo Cigno
Matthias Lott
Steven Low
Steven Lumetta
Sasa Lygizou
Wenchao Ma
Sridhar Machiraju
Gerald (Chip) Maguire
Carlo Majorani
Szabolcs Malomsoky
Stefan Mangold
Josep Mangues-Bafalluy
Sotiris Maniatis
Petteri Mannersalo
Costa Manuel
Peter Marbach
Gianluca Mardente
Raymond Marie
Ruediger Martin
Gaia Maselli
Xavier Masip-Bruin
Lorne Mason
Luisa Massari
Hiroyuki Masuyama
Ketan Mayer-Patel
Gary McAlpine
Daniel McClary
Steve McLaughlin
Janise McNair
Marco Mellia
Tommaso Melodia
Michael Menth
Michela Meo
Vivek Mhatre
Gyorgy Miklos
Fabio Milan
Jens Milbrandt
Dragan Milic
Geyong Min
Chen Ming
Enzo Mingozzi
Jelena Misic
Vojislav Misic
Vishal Misra
Michel Misson
Keith Mitchell
Is-Haka Mkwawa
Shantidev Mohanty
Prasant Mohapatra
Ugo Monaco
Edmundo Monteiro
Uday Mudoi
Biswanath Mukherjee
Alaa Muqattash
Amy Murphy
Luca Muscariello
Hedi Nabli
Keisuke Nakano
Sanjiv Nanda
Saishankar Nandagopalan
Ioanis Nikolaidis
Raja Nimmelapelli
Bobby Ninan
Peng Ning
Cristina Nita-Rotaru
Zhisheng Niu
Dusit Niyato
Koji Nonobe
Alessandro Nordio
Ilkka Norros
Antonio Nucci
Stephan Olariu
Ruy Oliveira
Lyndon Ong
Phoemphun Oothongsap
Jitendra Padhye
Sergio Palazzo
Jianping Pan
Antonis Panagakis
Ioannis Papadimitriou
Christos Papadopoulos
Symeon Papavassiliou
Sarantis Paskalis
Nikos Passas
Giovanni Pau
Jon Peha
Luciana Pelusi
Xue-hai Peng
Charles Perkins
Harry Perros
Daniele Perucchini
Marinella Petrocchi
Gian Pietro Picco
Chiara Piglione
Paulo Pinto
Andreas Pitsillides
Roman Pletka
Christos Politis
George Polyzos
Dario Pompili
Ana Pont
Adrian Popescu

Ramon Puigjaner
Guy Pujolle
Vishwas Puttasubbappa
He Qi
Kimmo Raatikainen
Pertti Raatikainen
Jan Rabaey
Andras Racz
S.V. Raghavan
Idris Rai
Parmesh Ramanathan
Rajesh Rao
Sanjay Rao
Erwin Rathgeb
Sylvia Ratnasamy
Miklós Aurél Rónai
Douglas Reeves
Hannu Reittu
Reza Rejaie
Fengyuan Ren
Xiangshi Ren
Julien Ridoux
Rui Rocha
Meysam Roodi
Meysam Roodi
Sean Rooney
Michele Rossi
Angelos Rouskas
George Rouskas
Antony Rowstron
Mohamed Saad
Tarek Saadawi
Amit Sahoo
Julio Sahuquillo
Amir Said
Leila Saidane
Aladdin Saleh
Theodoros Salonidis
Susana Sargento
Saswati Sarkar
Jahangir Sarker
Stefan Saroiu
Carla Savage
Amir Sayegh
Sergio Sánchez-López
Nicola Scalabrino
Matthias Scheidegger
Enrico Schiattarella
Gregor Schiele
Pegah Seddighian
Kari Seppänen
Renué Serral-Graciá
Neel Shah
Nirmala Shenoy
Hanif Sherali
Matthias Siebert
Arne Simonsson
Mayank Singhal
Harry Skianis
Dimitris Skyrianoglou
Paul Smith
Jungmin So
Katayoun Sohrabi
Hesham Soliman
Joo-Han Song
Mohit Sood
Augustin Soule
Liew Soung Chang
Otto Spaniol
Ashwin Sridharan
Matthias Stallmann
Vladica Stanisic
Ioannis Stavrakakis
Lothar Stibor
Ivan Stojmenovic
Aaron Striegel
Marinos Stylianou
Weilian Su
Xiao Su
Kamala Subramaniam
Kun Sun
Karthikeyan Sundaresan
Chi Wan Sung
Arak Sutivong
Tamas Suto
Violet Syrotiuk
Takuji Tachibana
Rahim Tafazolli
Christopher Taggart
Akira Takahashi
Yutaka Takahashi
Hideaki Takusagawa
Zhangxi Tan
Xiaofeng Tao
Alessandro Tarello
Camillo Taylor
Daniele Tessera
Vijay Tewari
George Thanos
Nigel Thomas
Asavari Thombare
David Thornley
Ali Tizghadam
Terence D. Todd
Alfredo Todini
Chai Keong Toh
C.K. Toh
Fujita Tomonori
Hajer Tounsi
Aleksandar Trifunovic
Anand Tripathi
Hua-Wen Tsai
David Tse
Athanasios Tsokanos
Giovanni Turi
Leonidas Tzevelekas
Raza Ul-Mustafa
Murat Uysal
Francesco Vacirca
Athanasios Vaios
Shahrokh Valaee
Andras Valko
Massimo Valla
Andrea Valletta
Constantinos Vassilakis
Vasos Vassiliou
Malathi Veeraraghavan
Yannis Viniotis
Neophytos Vlotomas
Mehmet Vuran
Sonia Waharte
Bernhard Walke
Bin Wang
ChunJiang Wang
Hao Wang

Jiahong Wang
Lan Wang
Li-Chun Wang
Pan Wang
Ping Wang
Quanhong Wang
Wenye Wang
Xudong Wang
Yao Wang
Yaya Wei
Erik Weiss
Yu Wen
Jeffrey Wieselthier
Lars Wolf
Adam Wolisz
Tung Chong Wong
Wing Shing Wong
De-An Wu
Haitao Wu
Kui Wu
Christos Xenakis
Jin Xiao
Yang Xiao
Haiyong Xie
Jiang Xie
Linlin Xie
Yufeng Xin
Yongqiang Xiong
Dan Xu
Jun Xu
Lisong Xu
Peng Xu
Sheng Xu
Guoliang Xue
Meeta Yadav
Bai Yan
Zhu Yanfeng
Fan Yang
Ji Yang
Jie Yang
Qu Yang
Shun-Ren Yang
Xinjie Yang
Yang Richard Yang
Marcelo Yannuzzi
Richard Yao
Jian Ye
Tao Ye
Changchuan Yin
Wang Ying
Liu Yingzhuang
Mika Ylianttila
S.J. Ben Yoo
Xiaohu You
Ossama Younis
Tang Youxi
Helen Yu-Zhang
Guangxin Yue
Theodore Zahariadis
Daniel Zappala
Harf Zatschler
James Zeidler
Keith Q.T. Zhang
Ping Zhang
Qian Zhang
Yongbing Zhang
Zhongwei Zhang
Ben Zhao
Dongmei Zhao
Lian Zhao
Kan Zheng
Karla Ziri-Castro
Joachim Zottl

Conference Sponsors

Bell Canada

Research in Motion

Sun Microsystems

Technical Committee 6, Communication Systems, of the
International Federation for Information Processing

Proceedings Sponsor

Software Telecommunications Group of the
Nortel Networks Institute, University of Waterloo

Table of Contents

Peer-to-Peer Networks

Topology-Aware Peer-to-Peer On-demand Streaming
Rongmei Zhang, Ali R. Butt, Y. Charlie Hu 1

The Scalability of Swarming Peer-to-Peer Content Delivery
Daniel Stutzbach, Daniel Zappala, Reza Rejaie 15

Leopard: A Locality Aware Peer-To-Peer System with No Hot Spot
Yinzhe Yu, Sanghwan Lee, Zhi-Li Zhang 27

PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications
David Hausheer, Burkhard Stiller 40

Performance of Internet Protocols I

Highly Responsive and Efficient QoS Routing Using Pre- and On-demand Computations Along with a New Normal Measure
Yanxing Zheng, Turgay Korkmaz, Wenhua Dou 53

An Architecture for Software-Based iSCSI: Experiences and Analyses
Annie Foong, Gary McAlpine, Dave Minturn, Greg Regnier, Vikram Saletore 65

Reorder Density (RD): A Formal, Comprehensive Metric for Packet Reordering
Nischal M. Piratla, Anura P. Jayasumana, Abhijit A. Bare 78

Batch Scheduling Algorithms for Optical Burst Switching Networks
Ayman Kaheel, Hussein Alnuweiri 90

Wireless Security

ECC Based Threshold Cryptography for Secure Data Forwarding and Secure Key Exchange in MANET (I)
Levent Ertaul, Weimin Lu 102

Mutual Authentication and Key Exchange Protocols with Anonymity Property for Roaming Services
Yixin Jiang, Chuang Lin, Xuemin Shen, Minghui Shi 114

An Energy-Efficient Image Representation for Secure Mobile Systems
Tim Woo, Catherine Gebotys, Sagar Naik 126

Efficient Clustering for Multicast Key Distribution in MANETs
Mohamed Salah Bouassida, Isabelle Chrisment, Olivier Festor 138

Network Security

Using Secure Coprocessors to Protect Access to Enterprise Networks
Haidong Xia, Jayashree Kanchana, José Carlos Brustoloni 154

Trusted Security Devices for Bandwidth Conservation in IPSec Environments
C.D. Mano, A. Striegel 166

LIPS: Lightweight Internet Permit System for Stopping Unwanted Packets
Changho Choi, Yingfei Dong, Zhi-Li Zhang 178

Victim-Assisted Mitigation Technique for TCP-Based Reflector DDoS Attacks
Basheer Al-Duwairi, G. Manimaran 191

Wireless Performance

Performance Analysis of the Uplink of a CDMA Cell Supporting Elastic Services
Gábor Fodor, Miklós Telek 205

Queue Analysis for Wireless Packet Data Traffic
Shahram Teymori, Weihua Zhuang 217

Modeling a Beacon Enabled 802.15.4 Cluster with Bidirectional Traffic
Jelena Mišić, Shairmina Shafi, Vojislav B. Mišić 228

Analysis of Windowing and Peering Schemes for Cache Coherency in Mobile Devices
Sandhya Narayan, Julee Pandya, Prasant Mohapatra, Dipak Ghosal 240

Network Service Support

Fast Scalable Robust Node Enumeration
Richard Black, Austin Donnelly, Alexandru Gavrilescu, Dave Thaler 252

gTrace: Simple Mechanisms for Monitoring of Multicast Sessions
Gísli Hjálmtýsson, Ólafur Ragnar Helgason, Björn Brynjúlfsson 264

Toward Feasibility and Scalability of Session Initiation and Dynamic QoS Provisioning in Policy-Enabled Networks
Kamel Haddadou, Yacine M. Ghamri-Doudane, Marc Girod-Genet, Ahmed Meddahi, Laurent Bernard, Gilles Vanwormhoudt, Hossam Afifi, Nazim Agoulmine 277

Modeling Soft State Protocols with SDL
Xiaoming Fu, Dieter Hogrefe 289

Network Modeling and Simulations

Inferring Traffic Burstiness by Sampling the Buffer Occupancy
Michel Mandjes, Remco van de Meent 303

Modeling Available Bandwidth for an Efficient QoS Characterization of a Network Path
Alexander Chobanyan, Matt W. Mutka, V.S. Mandrekar, Ning Xi 316

Reducing Large Internet Topologies for Faster Simulations
V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J-H. Cui, A.G. Percus 328

Dimensioning the Contention Channel of DOCSIS Cable Modem Networks
J. Lambert, B. Van Houdt, C. Blondia 342

Wireless LANs

CONTI: Constant-Time Contention Resolution for WLAN Access
Zakhia G. Abichar, J. Morris Chang 358

Efficient 3G/WLAN Interworking Techniques for Seamless Roaming Services with Location-Aware Authentication
Minsoo Lee, Gwanyeon Kim, Sehyun Park, Sungik Jun, Jaehoon Nah, Ohyoung Song 370

A New per-Class Flow Fixed Proportional Differentiated Service for Multi-service Wireless LAN
Meng Chang Chen, Li-Ping Tung, Yeali S. Sun, Wei-Kuan Shih 382

Reservation and Grouping Stations for the IEEE 802.11 DCF
Yang Xiao, Haizhon Li, Kui Wu, Kin K. Leung, Qiang Ni 395

Optical Networking

Path Switching in OBS Networks
Li Yang, George N. Rouskas 406

A Priority-Aware Protection Technique for Quality of Service Enabled WDM Networks
W. Fawaz, F. Martignon, K. Chen, G. Pujolle 419

Wavelength Rerouting in Survivable WDM Networks
Yingyu Wan, Weifa Liang 431

Performance of Internet & Web Applications

Performance of Server Selection Algorithms for Content Replication Networks
David Starobinski, Tao Wu 443

Local Utility Aware Content Replication
Nikolaos Laoutaris, Orestis Telelis, Vassilios Zissimopoulos, Ioannis Stavrakakis 455

Improving Network Convergence Time and Network Stability of an OSPF-Routed IP Network
Amir Siddiqi, Biswajit Nandy 469

Ad Hoc Networks I

Non-cooperative Forwarding in Ad-Hoc Networks
Eitan Altman, Arzad A. Kherani, Pietro Michiardi, Refik Molva 486

Efficient Broadcasting in Ad Hoc Networks Using Directional Antennas
Fei Dai, Jie Wu 499

Trellis-Based Virtual Regular Addressing Structures in Self-organized Networks
Julien Ridoux, Anne Fladenmuller, Yannis Viniotis, Kavé Salamatian 511

Adaptive Networking

Network of Shortcuts: An Adaptive Data Structure for Tree-Based Search Methods
Andrea Bergamini, Lukas Kencl 523

Multipath Routing Algorithms for Congestion Minimization
Ron Banner, Ariel Orda 536

Improving TCP in Wireless Networks with an Adaptive Machine-Learnt Classifier of Packet Loss Causes
Ibtissam El Khayat, Pierre Geurts, Guy Leduc 549

Performance of Internet Protocols II

A Multiple Time-Scale Model for TCP Bandwidth Sharing Under User Heterogeneity
Dirk Abendroth, Hans van den Berg, Michel Mandjes 561

A Multizone Pipeline Cache for IP Routing
Soraya Kasnavi, Paul Berube, Vincent C. Gaudet, José Nelson Amaral 574

Aggregated Aggressiveness Control on Groups of TCP Flows
Soohyun Cho, Riccardo Bettati 586

Radio Resource Management

Multi-level Dynamic Guard Channels for Priority Access in Cellular Systems
Tung Chong Wong, Jon W. Mark, Kee Chaing Chua 598

Slotted Aloha with Priorities and Random Power
E. Altman, D. Barman, A. Benslimane, R. El Azouzi 610

Downlink Packet Scheduling with Minimum Throughput Guarantee in TDD-OFDMA Cellular Network
Young Min Ki, Eun Sun Kim, Sung Il Woo, Dong Ku Kim 623

Internet Routing

Making IGP Routing Robust to Link Failures
Ashwin Sridharan, Roch Guérin 634

Interdomain Ingress Traffic Engineering Through Optimized AS-Path Prepending
Ruomei Gao, Constantinos Dovrolis, Ellen W. Zegura 647

Describing and Simulating Internet Routes
Jérémie Leguay, Matthieu Latapy, Timur Friedman, Kavé Salamatian 659

A Growth-Based Address Allocation Scheme for IPv6
Mei Wang 671

Queuing Models

Are Multiple Descriptions Better Than One?
György Dán, Viktória Fodor, Gunnar Karlsson 684

An Approximate Queueing Model for Limited-Range Wavelength Conversion in an OBS Switch
Vishwas S. Puttasubbappa, Harry G. Perros 697

On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks
Saurabh Tewari, Leonard Kleinrock 709

A Novel Direct Upper Approximation for Workload Loss Ratio in General Buffered Systems
József J. Bíró, András Gulyás, Zalán Heszberger 718

Network Monitoring, Measurement and Profiling

PISA: Automatic Extraction of Traffic Signatures
Parminder Chhabra, Ajita John, Huzur Saran 730

FPL-3: Towards Language Support for Distributed Packet Processing
Mihai Lucian Cristea, Willem de Bruijn, Herbert Bos 743

Efficient Deployment of Honeynets for Statistical and Forensic Analysis of Attacks from the Internet
Stephan Riebach, Erwin P. Rathgeb, Birger Toedtmann 756

Measuring Round Trip Times to Determine the Distance Between WLAN Nodes
André Günther, Christian Hoene 768

Network Management Systems

End-to-End Asymmetric Link Capacity Estimation
Ling-Jyh Chen, Tony Sun, Guang Yang, M.Y. Sanadidi, Mario Gerla 780

Impact of Resource Sharability on Dual Failure Restorability in Optical Mesh Networks
Chadi Assi, Wei Huo, Abdallah Shami 792

Efficient Distributed Solution for MPLS Fast Reroute
Dongmei Wang, Guangzhi Li 804

Pricing for Heterogeneous Services at a Discriminatory Processor Sharing Queue
Yezekael Hayel, Bruno Tuffin 816

Sensor and Ad Hoc Networks

Joint Sensor Selection and Data Routing in Sensor Networks
Ozgur Ercetin, Ozgur Gurbuz, Kerem Bulbul, Aylin Aksu 828

Peer Collaboration in Wireless Ad Hoc Networks
Lin Cai, Jianping Pan, Xuemin Shen, Jon W. Mark 840

Fireworks: An Adaptive Group Communications Protocol for Mobile Ad Hoc Networks
Lap Kong Law, Srikanth V. Krishnamurthy, Michalis Faloutsos 853

An Optimized TCP for Internet Access of Vehicular Ad Hoc Networks
Marc Bechler, Sven Jaap, Lars Wolf 869

Overlay Multicast

Cooperative Failure Detection in Overlay Multicast
Mengkun Yang, Zongming Fei 881

Eliminating Bottlenecks in Overlay Multicast
Min Sik Kim, Yi Li, Simon S. Lam 893

TOMA: A Viable Solution for Large-Scale Multicast Service Support
Li Lao, Jun-Hong Cui, Mario Gerla 906

Quality of Service

The Burstiness Behavior of Regulated Flows in Networks
Yu Ying, Ravi Mazumdar, Catherine Rosenberg, Fabrice Guillemin 918

A Credit-Based Active Queue Management (AQM) Mechanism to Achieve Fairness in the Internet
Gwyn Chatranon, Miguel A. Labrador, Sujata Banerjee 930

Optimizing End-to-End Throughput for Data Transfers on an Overlay-TCP Path
Pradnya Karbhari, Mostafa Ammar, Ellen Zegura 943

Wireless Scheduling

Randomized Coverage-Preserving Scheduling Schemes for Wireless Sensor Networks
Chong Liu, Kui Wu, Valerie King 956

Dynamic Scheduling for Scalable Media Transmission over cdma2000 1xEV-DO Broadcast and Multicast Networks
Kyungtae Kang, Jinsung Cho, Yongwoo Cho, Heonshik Shin 968

Proxy-Assisted Scheduling for Energy-Efficient Multimedia Streaming over Wireless LAN
Fan Zhang, Samuel T. Chanson 980

Improving Multicast Communication

Encodings of Multicast Trees
Vijay Arya, Thierry Turletti, Shivkumar Kalyanaraman 992

Efficient Bandwidth Guaranteed Restoration Algorithms for Multicast Connections
William Lau, Sanjay Jha, Suman Banerjee 1005

Scheduling Uplink Bandwidth in Application-Layer Multicast Trees
Sridhar Srinivasan, Ellen Zegura 1018

Traffic Management and Engineering I

A Novel Packet Marking Function for Real-Time Interactive MPEG-4 Video Applications in a Differentiated Services Network
Shane O'Neill, Alan Marshall, Roger Woods 1031

On Multipath Routing with Transit Hubs
A. Sen, B. Hao, B.H. Shen, S. Murthy, S. Ganguly 1043

Comparison of Border-to-Border Budget Based Network Admission Control and Capacity Overprovisioning
Ruediger Martin, Michael Menth, Joachim Charzinski 1056

Ad Hoc Networks II

RIDA: Robust Intrusion Detection in Ad Hoc Networks
Dhanant Subhadrabandhu, Saswati Sarkar, Farroq Anjum 1069

Self-configurable Key Pre-distribution in Mobile Ad Hoc Networks
Claude Castelluccia, Nitesh Saxena, Jeong Hyun Yi 1083

Efficient Use of Route Requests for Loop-Free On-demand Routing in Ad Hoc Networks
Hari Rangarajan, J.J. Garcia-Luna-Aceves 1096

Traffic Management and Engineering II

Penalty Shaper to Enforce Assured Service for TCP Flows
Emmanuel Lochin, Pascal Anelli, Serge Fdida 1108

QoS Dynamic Routing in Content Delivery Networks
Krzysztof Walkowiak 1120

Exploiting Traffic Localities for Efficient Flow State Lookup
Tao Peng, Christopher Leckie, Kotagiri Ramamohanarao 1133

The Interaction of Forward Error Correction and Active Queue Management
Tigist Alemu, Yvan Calas, Alain Jean-Marie 1146

Mobility Management

ELIP: Embedded Location Information Protocol
Farid Benbadis, Marcelo Dias de Amorim, Serge Fdida 1156

Coordinated Interaction Using Reliable Broadcast in Mobile Wireless Networks
Theodore L. Willke, Nicholas F. Maxemchuk 1168

Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks
Khaled A. Harras, Kevin C. Almeroth, Elizabeth M. Belding-Royer . . . 1180

Application Signal Threshold Adaptation for Vertical Handoff in Heterogeneous Wireless Networks
Ben Liang, Ahmed H. Zahran, Aladdin O. M. Saleh 1193

Bandwidth Management

Justice: Flexible and Enforceable Per-Source Bandwidth Allocation
Jakob Eriksson, Michalis Faloutsos, Srikanth Krishnamurthy 1206

A Novel Approach for Transparent Bandwidth Conservation
David Salyers, Aaron Striegel 1219

Efficient Bandwidth Sharing in Bus-Based Optical Access Networks
Andre-Luc Beylot, Nizar Bouabdallah, Guy Pujolle 1231

CDMA

Cross-Layer Radio Resource Allocation in Packet CDMA Wireless Mobile Networks with LMMSE Receivers
Fei Yu, Vikram Krishnamurthy 1243

Stabilization of Contention-Based CDMA Ranging Channel in Wireless Metropolitan Area Networks
Jeong-Jae Won, Choong-Ho Cho, Hyong-Woo Lee, Victor C.M. Leung 1255

Load Balancing and Relaying Framework in TDD W-CDMA Multi-hop Cellular Networks
Y. Hung Tam, Ahmed M. Safwat, Hossam S. Hassanein 1267

Wireless Resource Management

Stochastic Decision-Based Analysis of Admission Control Policy in Multimedia Wireless Networks
Nidal Nasser 1281

Improving the Performance of Rate Adaptation Schemes in Heterogeneous Wireless Networks
P. Stathopoulos, L. Sarakis, N. Mitrou 1297

Stochastic Admission Control for Quality of Service in Wireless Packet Networks
Majid Ghaderi, Raouf Boutaba, Gary W. Kenward 1309

Posters

Extending the Birkhoff-von-Neumann Switching Strategy to Multicast Switches
Jay Kumar Sundararajan, Supratim Deb, Muriel Médard 1321

A New TCP-Friendly Rate Control Algorithm for Scalable Video Streams
Jinyao Yan, Martin May, Kostas Katrinis, Bernhard Plattner 1326

802.11 Link Interference: A Simple Model and a Performance Enhancement
Hoon Chang, Vishal Misra 1330

Impact of Routing Lags on Internet Routing Failures
Feng Wang, Lixin Gao 1334

Control and Forwarding Plane Interaction in Distributed Routers
Markus Hidell, Peter Sjödin, Olof Hagsand 1339

Performance Analysis of 802.11 WLANs Under Sporadic Traffic
M. Garetto, C.-F. Chiasserini 1343

Virtual Multi-Homing: On the Feasibility of Combining Overlay Routing with BGP Routing
Zhi Li, Prasant Mohapatra, Chen-Nee Chuah 1348

Measurement of Rain Attenuation of Microwaves at 12.25GHz in Korea
Dong You Choi 1353

Scalable Route Selection for IPv6 Multihomed Sites
Cédric de Launois, Steve Uhlig, Olivier Bonaventure 1357

On the Efficient Resource Allocation for High Quality Video Streams and FTP Traffic over Next Generation Wireless Networks
Polychronis Koutsakis, Spyros Psychis 1362

A Self-organizing Routing Scheme for Random Networks
Thomas Fuhrmann 1366

A Novel Extension for On-demand Routing Protocol in Event-Driven Sensor Networks
Dong-Hyun Chae, Kyu-Ho Han, Kyung-Soo Lim, Sun-Shin An 1370

Equivalent Gibbs Duhem Law in Network Queue
Sandeep Sudeep, Joseph Y. Hui 1374

Bandwidth Sharing of Low Priority Services for Efficient Call Admission Control in Cellular Networks
Kyungkoo Jun, Seokhoon Kang 1379

COPACC: A Cooperative Proxy-Client Caching System for On-demand Media Streaming
Alan T.S. Ip, Jiangchuan Liu, John C.S. Lui 1383

A Rule-Based Approach for Consistent Proportional Delay Differentiation
Jianbin Wei, Cheng-Zhong Xu 1388

Extended Dominating Set in Ad Hoc Networks Using Cooperative Communication
Jie Wu, Mihaela Cardei, Fei Dai, Shuhui Yang 1393

INTERMON: An Architecture for Inter-domain Monitoring, Modelling and Simulation
Elisa Boschi, Salvatore D'Antonio, Paul Malone, Carsten Schmoll . . . 1397

An UNA-Based Approach to Support Mobility in the Internet
Mahnhoon Lee, Yongik Yoon 1401

QoS Scalable Tree Aggregation
Joanna Moulierac, Alexandre Guitton 1405

Supermedia Transport for Teleoperations over Overlay Networks
Zhiwei Cen, Matt W. Mutka, Danyu Zhu, Ning Xi 1409

A Simple and Fast Algorithm for Bluetooth Network Formation
Yuh-Jzer Joung, Geng-Dian Hwang 1413

A Node Cooperative ARQ Scheme for Wireless Ad-Hoc Networks
Mehrdad Dianati, Xinhua Ling, Sagar Naik, Xuemin Shen 1418

SWATCH: a StepWise AdapTive Clustering Hierarchy in Wireless Sensor Networks
Quanhong Wang, Kenan Xu, Hossam Hassanein, Glen Takahara 1422

Design and Performance Evaluation of WDM/TDMA-Based MAC Protocol in AWG-Based WDM-PON
Kyeong-Eun Han, Yang He, Seung-Hyun Lee, Biswanath Mukherjee, Young-Chon Kim 1426

A Backup Tree Algorithm for Multicast Overlay Networks
Torsten Braun, Vijay Arya, Thierry Turletti 1430

Tradeoffs for Web Communications Fast Analysis
Olivier Paul, Jean-Etienne Kiba 1435

Benefits of Multiple Battery Levels for the Lifetime of Large Wireless Sensor Networks
Mihail L. Sichitiu, Rudra Dutta 1440

Enhanced Data Services in GPRS Networks via Auction-Based Prices for Admission
Saravut Yaipairoj, Fotios C. Harmantzis 1445

Achieving Stability and Fairness in Mobile Ad Hoc Networks
Ahmed M. Mahdy, Jitender S. Deogun, Jun Wang 1449

A Novel Method for SCTP Load Sharing
Andreas Jungmaier, Erwin P. Rathgeb 1453

Attaining VoIP–Grade QoS via Deflection: A Buffer Space Tradeoff Study
André Muezerie, Ioanis Nikolaidis, Pawel Gburzyński 1457

Applying the DiffServ Model to a Resilient Packet Ring Network
Fredrik Davik, Stein Gjessing 1461

Maximizing the Number of Connections in Multifiber WDM Chain, Ring and Star Networks
Katerina Potika 1465

A Hierarchical Secure Ring-Oriented Multicast Protocol over Mobile Ad Hoc Network
Choong Seon Hong, Yubai Yang 1471

Performance Analysis of a Multimedia CDMA System Using Dynamic Rate Control
Seung Sik Choi 1475

Author Index 1479

Topology-Aware Peer-to-Peer On-demand Streaming

Rongmei Zhang, Ali R. Butt, and Y. Charlie Hu

Purdue University, West Lafayette IN 47907, USA
{rongmei, butta, ychu}@purdue.edu

Abstract. In this paper, we consider large-scale high-bandwidth on-demand media streaming in a dynamic and heterogeneous environment. We present MetaStream, a scalable and distributed content discovery protocol that enables clients across the Internet to self-organize into a topology-aware overlay network, in which they can "cache and relay" a stream among nearby peers. We present the design and implementation of MetaStream and show experimental results obtained in a large-scale emulated environment. Our evaluation shows that MetaStream distributes relaying load among the clients in a topology-aware manner, imposing low link cost to the underlying network and low streaming load on the media server. MetaStream is also highly resilient to node or network failures and is capable of quickly recovering the streaming quality at clients even at high failure rates.

Keywords: Peer-to-peer, on-demand streaming, overlay networks, topology-awareness.

1 Introduction

In this paper, we consider the problem of on-demand media streaming to a large set of clients spread across the Internet. Compared to real-time media streaming or simultaneous file downloading, on-demand streaming is distinct in two aspects: (1) asynchrony, as clients may request for the media at different times; and (2) non-sequentiality, as different clients may access the media starting at different parts of the media object.

There exist two approaches to on-demand streaming. The first approach adopts IP multicast to serve multiple requests by a single stream. Due to the synchronous nature of multicast, clients either wait for the next scheduled multicast session at the cost of some start-up delay [9, 12], or participate in more than one sessions simultaneously [15, 8]. The second approach exploits caching of media objects at the proxies [20, 3, 21], or at the clients [23, 7, 14, 5, 6]. The media can be retrieved from the caches instead of from the streaming server.

The client-side "cache and relay" scheme has several major advantages. First, it is an application-level solution without any assumptions about the underlying network (i.e., IP multicast), and it does not require the deployment of specialized proxies. Receivers cache a small portion of the media content after playback and they collectively form a cooperative overlay network by streaming from each other's cache. Second, the capacity of the streaming overlay scales with the population of clients. It has been shown that this "grassroot" approach can defeat IP-multicast-based solutions in terms of server-side bandwidth saving [5]. Since clients also contribute streaming bandwidth to the

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1–14, 2005.

streaming overlay, it has the self-scaling property of peer-to-peer file sharing systems. Third, each client can select the streaming source based on various QoS requirements. When there exist more than one candidates in the overlay network, the client can choose the one with the best performance metric, e.g., the lowest delivery delay.

A key challenge for the client-side "cache and relay" approach is *how to locate available streaming sources in the overlay network, given that clients may be widely dispersed over the Internet and they can leave or fail at any time.* This calls for a content discovery service for streaming information update and lookup. A straightforward solution is to deploy a centralized server that keeps track of all the streaming sessions. Obviously this solution suffers from the bottleneck and single point of failure at the centralized server. In [5], the content discovery service is implemented by a number of clients acting as discovery servers. Specifically, each streaming session is registered with the content discovery service by contacting a subset of discovery servers. Likewise, each request is also mapped to a subset of the discovery servers. A client can find eligible streaming sources by querying the content discovery service. However, the mapping between streaming sessions or requests to discovery servers is performed through inconsistent hashing, i.e., the mapping is dependent on the number of available servers.

In this paper, we propose MetaStream, a fully decentralized and adaptive protocol for content discovery in large-scale high-bandwidth streaming. Since the volume and the physical distribution of streaming requests are usually beyond estimation, it is difficult to pre-determine the scale and the deployment of a content discovery service. In MetaStream, all clients participate in the content discovery service and take on the responsibility of finding and choosing streaming sources without any requirement for external configuration or management. MetaStream is also adaptive to client population and network condition dynamics, and streaming paths can be refined over time to maintain and improve the streaming quality at clients. When a failure happens at a client or in the network, MetaStream can relocate the streaming source and restore the streaming session quickly. Moreover, MetaStream is oblivious to the encoding of the media object. MetaStream can be used with any data encoding schemes such as layered encoding [21] or multiple descriptor encoding (MDC) [10] to address the resource heterogeneity of streaming clients.

In addition, MetaStream also aims to minimize the streaming cost at the backbone network. Due to the high streaming rates of media objects, the streaming overlay should be network bandwidth efficient, and should avoid long streaming paths between clients. For the content discovery service, this translates into *topology-awareness*: for each streaming request, if there are multiple candidates capable of providing the media object, the one that is closest in terms of network distance should be chosen. MetaStream has inherent topology-awareness by clustering clients based on the network distances between them, and a streaming request is always satisfied by a nearby source, if it exists. Topology-awareness also contributes to the scalability of MetaStream.

In summary, MetaStream provides an efficient distributed content discovery service that allows clients to locate streaming sources in a topology-aware manner. As a result, clients organize themselves into a topology-aware overlay network. We have developed a prototype implementation of MetaStream and have performed an extensive evaluation of MetaStream running over 1,000 clients in an emulated Internet topology consisting of 5,050 routers. The experimental results show that MetaStream can satisfy client streaming requests with only minimal server-side bandwidth consumption and at the same time incurs low network link cost.

The rest of the paper is organized as follows. Section 2 gives a brief background on the "cache and relay" approach to on-demand streaming. Section 3 presents MetaStream, a topology-aware content discovery protocol for media streaming. Section 4 discusses the methodology for evaluating MetaStream and Section 5 provides detailed evaluation results obtained from a large-sale emulated environment. Finally, Section 6 reviews related work and Section 7 draws concluding remarks.

2 Preliminaries

In the following, we give a brief overview of the client-side "cache and relay" scheme [5, 6, 14] for on-demand streaming.

For generality, we assume that the media object consists of one or more stripes, depending on if data encoding is applied. The object is originally provided by server C_0. The request from client $C_i (i = 0, \ldots, n)$ is characterized by the time T_i that C_i starts streaming and the offset S_i in the stream where the streaming begins. The achievable streaming quality at C_i is bounded by the input bandwidth. Each client also maintains a small cache of the media after playback and we denote the cache length as W_i. A later client C_j can stream from client C_i if their requests satisfy the following condition: $(T_i - S_i) < (T_j - S_j) < (T_i - S_i) + W_i$. The amount of stripes that a client C_i can relay to others is limited by its own output bandwidth. Any stripes that cannot be provided by the caches of other clients can be retrieved from the original server C_0.

A simple example of "cache and relay" is shown in Figure 1, where a stream is encoded into 3 stripes. Suppose that each client maintains a cache of 5 minutes and each starts the streaming from the beginning. The first client C_1 receives 2 stripes directly from the server C_0. The second client C_2 can retrieve the first 2 stripes from client C_1, but the third stripe has to be streamed from the server. When the third client C_3 arrives, its streaming request can be satisfied by client C_2.

Figure 2 gives an example of "topology-awareness" by showing a simplified network of routers connecting the streaming server and clients in Figure 1. If we assume

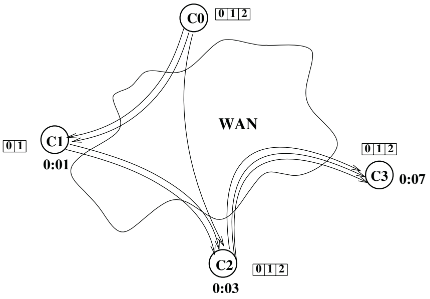


Fig. 1. Cache and relay

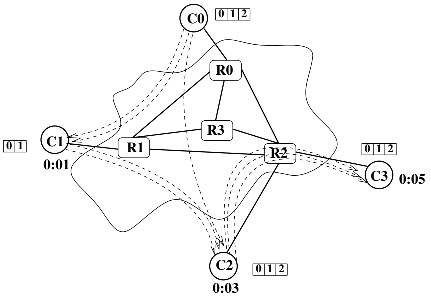


Fig. 2. Topology-aware cache and relay

client C_3 arrives 2 minutes after client C_2 and 4 minutes after client C_1, it is able to stream the first two stripes from either C_1 or C_2. However, it is more beneficial from the perspective of network bandwidth consumption to select C_2 since C_2 is only 2 hops away and C_1 is at least 3 hops away.

3 MetaStream

In this section, we present the design of MetaStream, a content discovery protocol for topology-aware on-demand streaming.

3.1 Topology-Based Clustering

In MetaStream, clients choose streaming sources based on the network distance. For this suppose, they self-organize into a dynamic hierarchy of clusters based on the network topology. In principle, any protocol for constructing a topology-aware hierarchy can be used. We use the NICE protocol [1] for the following reasons: (1) The source code is publicly available; (2) The hierarchy can be built and maintained with very low overhead; (3) The hierarchy is adaptable to membership and network condition changes.

Due to page limitation, we omit the details of the NICE protocol. An example hierarchy of topology-based clusters as can be built using NICE is shown in Figure 3. Cluster leaders (i.e., cluster centers) of each level join the next higher level clusters, e.g., C_1, C_2 and C_3 of level-0 join level-1. By using a hierarchy of clusters, MetaStream can scale to a large number of clients.

3.2 Topology-Aware Content Discovery

To facilitate discovering nearby available streaming sources, a content discovery service is built on top of the topology-aware hierarchy.

Streaming State Registration and Aggregation. First, streaming clients with available output bandwidth register with the content discovery service. Specifically, each client sends the information about the stripes being received and thus cached locally to its bottom level cluster leader. After aggregating the information from all the clients in the cluster, the bottom cluster leader sends a summary registration to the next higher level cluster leader. This registration process ends at the hierarchy root, which is the leader of the top level cluster.

The information presented to cluster leaders during registration is light-weight. Recall from Section 2 that client C_j can stream from client C_i if their requests satisfy $(T_i - S_i) < (T_j - S_j) < (T_i - S_i) + W_i$. Therefore, it suffices for client C_i to provide a range $(T_i - S_i, T_i - S_i + W_i)$ in the registration. Aggregation within each cluster is simply performed as a disjunction of all received registrations $\bigcup_i (T_i - S_i, T_i - S_i + W_i)$.

An example of aggregation is shown in Figure 3. Each stripe of the stream is registered and aggregated separately. For clarity, only the aggregation for one stripe is shown in Figure 3. As a client may belong to multiple levels in the hierarchy, it may be responsible for receiving registrations and performing aggregations at each of these levels. If the hierarchy is created using the NICE protocol, each client maintains no more than $O(\log N)$ registrations from other clients.

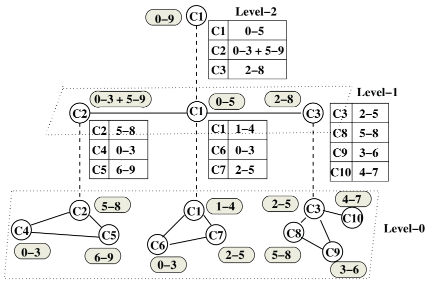


Fig. 3. Aggregation of cache state

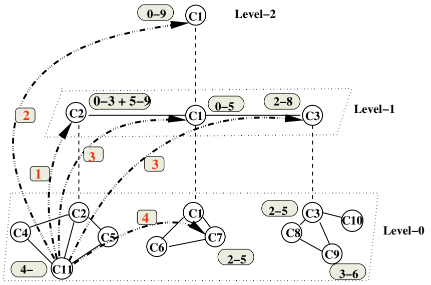


Fig. 4. Resolving streaming request

Buffer registration is performed periodically and asynchronously by each client. Registration information is maintained as soft-state. The soft-state approach keeps registrations up-to-date as the hierarchy is undergoing all kinds of possible changes, such as when clients join and leave, clusters split or merge, and clients crash without notifying other members. In addition, MetaStream also invokes on-demand registration to react to changes in the composition of the stripes in a client's cache and its output bandwidth availability.

Resolving Request for a Single Stripe. A client request for a streaming stripe is resolved by querying the content discovery service. Only the timing of the request, in the form of $(T_j - S_j)$, needs to be presented. At each step of resolving the request, potential streaming sources are identified by checking $(T_j - S_j)$ against the aggregated cache registration $\bigcup_i (T_i - S_i, T_i - S_i + W_i)$.

Figure 4 shows an example of resolving a streaming request based on the aggregation example in Figure 3. Suppose that a new client C_{11} wants to find a streaming source for stripe x. A request is first sent to the level-0 cluster leader C_2 (step 1). Generally, a request is forwarded up the hierarchy until it receives a successful response (step 2). Since higher levels have a wider view of the network, the request is more likely to be resolved. A successful response includes those next lower level cluster members that can further resolve the request. If the request is received at the root and the root is not aware of any other clients that can relay stripe x from their caches, server C_0 can accept the request and open a streaming session for the requested stripe.

After receiving one successful reply, the resolving process starts traversing the hierarchy downward (step 3), as it zooms in towards the closest streaming source. When the resolving process arrives at the bottom level and the closest candidate is determined, e.g., to be client C_7, client C_7 accepts the request if it confirms that it has enough output bandwidth remaining to relay stripe x to C_{11}; at this point the request is fully resolved.

Due to propagation delay along the hierarchy and the fact that aggregation partially relies on periodical refreshments, at some point in resolving the request, it may happen that client C_1 does not contain any matching record or client C_7 does not have enough remaining output bandwidth to accept C_{11}'s streaming request. When the closest candidate fails to further resolve the request, clients can continue to contact the next closest candidate. MetaStream also enables clients to back-trace to the previous levels of the hierarchy. This can be made possible by bookkeeping (e.g., using a stack) the results at each step of resolving the request.

Resolving Requests for Multiple Stripes. In the descriptions of the content discovery service above, we consider only one stripe of the stream and make no assumption about the actual encoding scheme. Whether multiple description [10] or layered encoding [21] is used, the stripes can be resolved sequentially. As soon as one stripe is resolved, the stream is renderable at the client; the streaming quality improves as more stripes are available. In multiple description encoding, clients can choose from different combinations given the total number of stripes. In fact, clients just need to specify the desired number of stripes without defining which stripes. In contrast, in layered encoding, the order of the stripes to be resolved is fixed. Therefore, under comparable circumstances, multiple description coding allows more flexibility in satisfying clients' desired streaming quality.

It is worth noting that the NICE hierarchy is only used for propagating control messages during streaming state registration and streaming source discovery, and it is not involved in the data streaming process once the streaming source is located.

3.3 Handling Failures

The NICE hierarchy is adaptive to node failures or network partitions through automatic reconfiguration. Similarly, MetaStream handles a streaming session failure due to the above reasons by relocating the streaming source, using the same content discovery protocol described in this section. If a failed client is able to continue relaying the stream from its cache, it can sustain the streaming until the cache is depleted, while those downstream clients re-locate alternative sources. Even if the failed client stops relaying immediately, it is unlikely to disrupt the entire streaming of downstream clients if multiple-stripe encoding is used; they would only experience temporary deterioration of the streaming quality.

3.4 Refining and Adapting the Streaming Overlay

The streaming overlay is dynamic as clients join and depart. This raises opportunities for optimizing streaming paths. If a stripe is currently provided by the server, the server load will be reduced if the stripe can be switched to another client. In addition, the streaming efficiency of the overlay can also be improved by attaching a stripe to a closer available streaming source. Both situations can be exploited by periodically attempting to re-attach selected stripes in a controlled manner, e.g., by querying an appropriate number of levels up the content discovery hierarchy.

So far MetaStream has only considered bandwidth constraints at the edge of the network, i.e., at the server and clients. In practice, internal links in the underlying network might become the bottleneck. MetaStream can be modified to take the available bandwidth of the streaming paths into consideration. For instance, the client can select the top few closest candidates during the content discovery, measure the throughput to each of them, and choose the one with the highest throughput. During the streaming, the network may not be able to deliver the throughput required at the client. The client can adapt by dropping the current connection and re-attaching to an alternative streaming source.

4 Experimental Methodology

In this section, we describe the experimental methodology for evaluating MetaStream.

4.1 MetaStream Implementation

We have implemented MetaStream as a stand-alone application in C++, on top of the NICE code (http://www.cs.umd.edu/ suman/research/myns/index.html). The NICE code exports an API that can be used to build applications on top of the NICE hierarchy. We also modified the core API to allow for socket communication. This modification is transparent to applications built on top of the core API. In total, our implementation of MetaStream consists of about 5500 lines of code.

4.2 ModelNet Emulator

We evaluated our MetaStream implementation in the ModelNet [24] IP emulation framework. Our ModelNet setup consisted of four machines, each with a 3.0GHz Pentium IV processor and 512MB RAM, interconnected with a Gigabit Ethernet switch. Two of these machines ran Free BSD 4.7, and served as the ModelNet core cluster, while the remaining two ran RedHat 9.0 and served to support the virtual nodes. A network topology generated by GT-ITM described below was emulated by the core routers and used to connect 1000 virtual nodes.

4.3 Network Model

The experiments are conducted using a network topology with 5050 routers generated by the GT-ITM [25] graph generator using the transit-stub model. The streaming server and 1000 clients are randomly attached to the stub routers.

To model the heterogeneity of the capacities of client nodes in the Internet, we categorize clients into 3 groups according to their bandwidth similarly as in [6]: (1) 50% clients have maximum total bandwidth of 128Kbps and they represent Modem/ISDN users; (2) 35% clients have maximum total bandwidth of 1Mbps and they have Cable Modem/DSL connections; (3) the rest 15% clients have maximum total bandwidth of 10Mbps and they have Ethernet connections. The inbound and outbound bandwidth are allocated from the total available bandwidth. We control the allocation using outbound/inbound bandwidth ratio r. The larger the ratio, the more bandwidth is contributed to streaming to other clients.

4.4 Data Model

The media object is encoded into 50 stripes using layered encoding, similarly as in [6]. All stripes are assumed to have an identical rate of 20Kbps, although in practice stripes can be encoded using different streaming rates. The total streaming rate is 1Mbps. The full length of the stream is 20 minutes (1200 seconds) and each client caches 100 seconds of received stripes after playback.

Streaming clients arrive according to the Poisson process and the average interarrival time is 10 seconds. Each client leaves after finishing the 20-minute streaming, and the simulation stops when the last client finishes streaming.

4.5 Protocols Evaluated

MetaStream considers both the server load and the network link cost. These two goals can conflict with each other. We are interested in whether the server load has to be compromised to achieve lower link cost. The problem of optimizing server load alone for layer-encoded on-demand streaming is NP-complete under client output bandwidth constraints [6]. In [6], a greedy algorithm is proposed and it can maximize the number of stripes that a new client can get from other clients at the moment of joining the streaming overlay. We term this greedy algorithm "Greedy:ServerLoad" and it represents the best effort towards minimizing server load to our knowledge. We implemented the "Greedy:ServerLoad" algorithm to run on the same emulator.

For comparison, we have also implemented a unicast-based on-demand streaming approach, in which each client simply streams all desired stripes from the server. Although this approach has obvious drawbacks such as overloading the server and the network links close to the server, it is widely used in the Internet today.

The NICE hierarchy used by MetaStream is maintained by *HeartBeat* messages within clusters and the default heartbeat interval is set to 3 seconds. We also set the interval for refreshing streaming cache state to 3 seconds for both MetaStream and the "Greedy:ServerLoad" scheme.

4.6 Performance Metrics

We compare the performance of various solutions using the following metrics: (1) *Path length* measures the average number of network-layer routing hops of the streaming paths. (2) *Link stress* measures the data traffic at the network links, averaged over all the links that have through data traffic. Together, link stress and path length give a complete picture of the network link cost of streaming. (3) *Server load* measures the bandwidth consumption at the streaming server. It reflects the effectiveness of the protocol in locating client-side streaming sources. (4) *Discovery delay* refers to the latency for a client to locate/relocate the streaming sources. (5) *Control overhead* comes from two sources in MetaStream: the overhead in constructing and maintaining the topology-aware hierarchy, and the overhead in updating and querying streaming information. All the control messages exchanged are counted in measuring the control overhead.

5 Experiment Results

We present the results from extensive experiments in this section. We first evaluate MetaStream in a typical setting and then study the effects of various system parameters.

5.1 Representative Scenario

As a representative scenario, the outbound/inbound bandwidth ratio is set to 1.0. This models the peer-to-peer spirit which requires each node to contribute as much output bandwidth as its input bandwidth consumption.

Path Length. Figure 5 shows the CDF of the average path length of all the layers received by each client. The diameter of the network topology is 20 hops. In MetaStream,

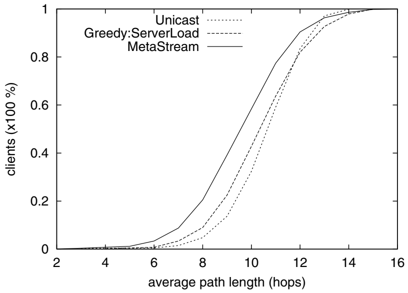


Fig. 5. Representative scenario: CDF of path length

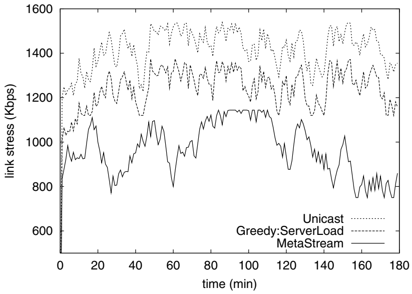


Fig. 6. Representative scenario: link stress

on average all the layers received at each client travel at most 10 hops (half of the network diameter) for about 58.3% of the clients, but for only about 42.8% clients using the "Greedy:ServerLoad" algorithm. In addition, for each layer, MetaStream reduces the average streaming path length by more than one hop, compared with the "Greedy:ServerLoad" algorithm (the result is not shown due to page limitation).

Link Stress. Shorter path length translates into potentially lower link stress on the underlying network, and this is confirmed in Figure 6, which shows the average streaming traffic over network links during the experiment. MetaStream induces about 10-20% less link stress compared with the "Greedy:ServerLoad" algorithm. The unicast approach generates much higher link stress than both the "Greedy:ServerLoad" algorithm and MetaStream, which can be explained by the high bandwidth consumption near the streaming server. The results on the streaming path length and network link stress show that by taking network topology into account, MetaStream can effectively reduce the streaming cost on the backbone network.

Server Load. Figure 7 shows the server load measured by the number of layers that the server has to stream. First, the "cache and relay" approach can significantly reduce the bandwidth demand at the streaming server. By using "cache and relay", MetaStream and the "Greedy:Server Load" algorithm bring server load down to below 1% of the total bandwidth demand by clients. The peak bandwidth consumption is only about 4Mbps (200 layers) at the server, which is equivalent to 4 clients with full quality streams. Even a desktop computer with Ethernet connection can sustain the server load in this scenario. Second, MetaStream achieves almost the same amount of server load as the "Greedy:ServerLoad" algorithm, which is originally designed for the purpose of minimizing server side bandwidth. This implies that MetaStream does not sacrifice server load for lower network cost and the "cache and relay" approach itself can effectively limit the server-side load. We have run both algorithms under various combinations of parameters and in all the situations that we have investigated, MetaStream achieves comparable server bandwidth savings as the "Greedy:ServerLoad" algorithm.

Discovery Delay. Figure 8 shows the CDF of the startup delay at the clients. The link delay generated by GT-ITM is used in calculating the delay in resolving streaming

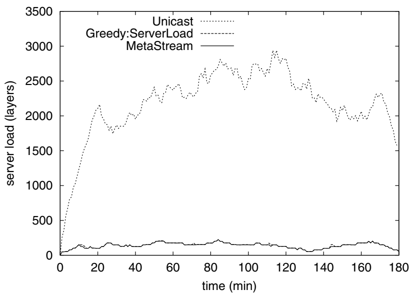


Fig. 7. Representative scenario: server load

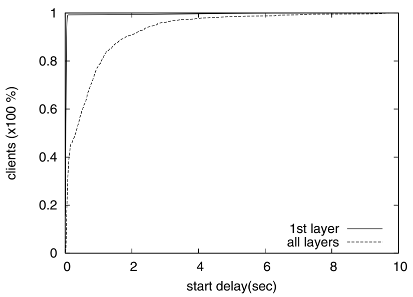


Fig. 8. Representative scenario: CDF of content discovery delay

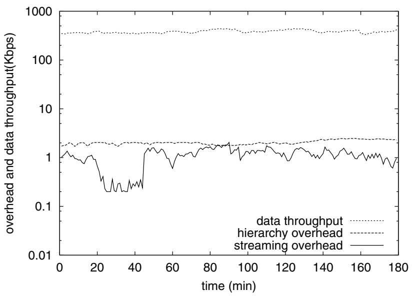


Fig. 9. Representative scenario: control overhead in MetaStream

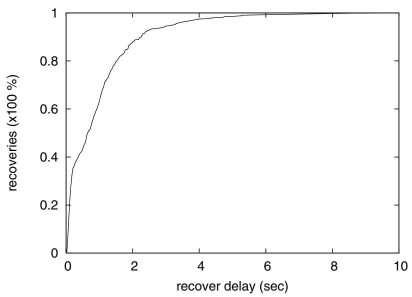


Fig. 10. CDF of failure recover delay

requests. Over 99% of clients experience less than 1 second delay in locating the first stripe and over 90% experience less than 2 seconds delay in locating all the stripes. The maximum delay is under 10 seconds for both data curves.

Control Overhead. The control overhead of MetaStream at clients is depicted in Figure 9. The overhead bandwidth accounts for both the incoming and outgoing control traffic at clients. The volume of control traffic from streaming related activities and from the NICE protocol are comparable. The control overhead induced by the NICE protocol remains stable as clients join and leave the streaming overlay. The overhead from MetaStream itself fluctuates slightly as the number of active streaming layers changes over time (as shown in Figure 7). Both overhead are two orders of magnitude less than the streaming data received at clients.

5.2 Effects of System Parameters

The performance of MetaStream varies with the following parameters:

- *Cache size:* Long buffers imply that more streaming requests can be satisfied by other clients (Figure 11). It also allows for more choices in selecting streaming sources and therefore better quality for streaming paths (Figure 12).

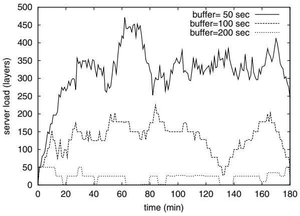


Fig. 11. Server load with varying cache length: outbound/inbound = 1.0, average arrival interval = 10 sec

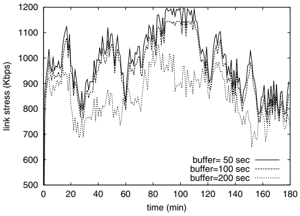


Fig. 12. Link stress with varying cache length: outbound/inbound = 1.0, average arrival interval = 10 sec

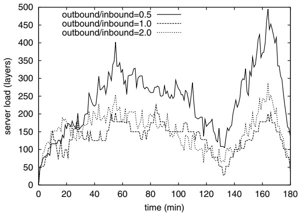


Fig. 13. Server load with varying outbound/inbound ratio: cache length = 100 sec, average arrival interval = 10 sec

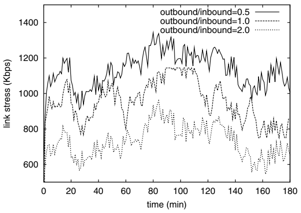


Fig. 14. Link stress with varying outbound/inbound ratio: cache length = 100 sec, average arrival interval = 10 sec

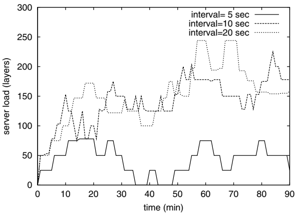


Fig. 15. Server load with varying client arrival rate: cache length = 100 sec, outbound/inbound = 1.0

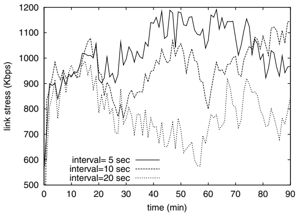


Fig. 16. Link stress with varying client arrival rate: cache length = 100 sec, outbound/inbound = 1.0

– *Client output bandwidth:* The availability of a client as a streaming source is determined not only by the timing dependency between streaming requests, but also by the output bandwidth of the client. With all the other factors fixed, more abundant output bandwidth enables a client to serve more other clients (Figure 13). This can also result in shorter streaming paths for some clients (Figure 14).

5.3 Scalability

A third parameter that can affect the performance of MetaStream is the client arrival rate. For a fixed streaming length, higher arrival rate translates into larger client population, and thus varying the arrival rate measures the scalability of MetaStream. On one hand, increased client population increases the aggregate bandwidth demand. On the other hand, it increases the opportunity for "cache and reply" from other clients instead of from the server. Figure 16 and Figure 15 show that MetaStream scales well with the client population, in terms of the bandwidth demand on both the server side and the underlying network.

5.4 Fault Recovery

In order to evaluate the failure recovery capability of MetaStream, we introduced random failures in our experiment of the representative scenario. Out of the 1000 clients,

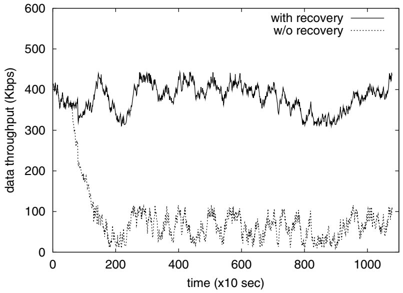


Fig. 17. Throughput at clients, with failures

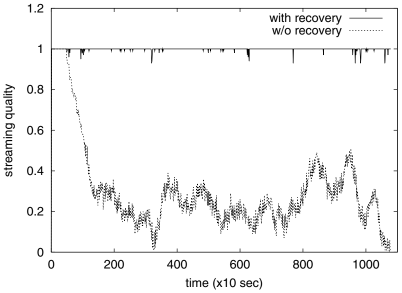


Fig. 18. Streaming quality at clients, with failures

30% fail before they finish the 20 minute streaming. Failures are randomly injected among the clients and for each failed client, the failure time is determined by a random value in between (1, 20) minutes.

We also assume that clients fail silently and stop streaming to all downstream clients immediately. Since the stream is encoded using the layer encoding scheme, once a layer is interrupted due to source failure, all higher layers become useless and are therefore also disconnected from their corresponding streaming sources. All disrupted layers will be recovered by the failure handling mechanism. The directly affected clients continue to stream to their own downstream receivers, and thus their caches can absorb some of the potentially cascading effects of upstream failures in the streaming overlay.

Figure 17 and Figure 18 show the streaming throughput received and the streaming quality experienced by an average client, with and without failure recovery, respectively. Client streaming quality is defined by the percentage of decodable layers out of all the desired layers. These results show that MetaStream can fully restore all the streaming sessions and clients are hardly affected by failures except a tiny delay for recovery. Meanwhile the server load remains roughly at similar levels as in failure-free situations (the result is omitted due to page limit). Figure 10 depicts the CDF of the delay to recover from streaming source failures, i.e., the time from when the streaming quality at the client declines due to source node failures to when the streaming quality fully recovers. The failure recovery delay is comparable with the startup delay shown in Figure 8.

6 Related Work

MetaStream is related to previous work in the area of on-demand streaming. There has been a rich body of work on on-demand streaming which roughly fall into two categories: multicast-based techniques [12, 15, 9, 8] and media caching [20, 3, 23, 4]. MetaStream is closely related to several recent work on client-side media caching [5, 6, 14]. Also related to our work is overlay-based multicast and streaming [13, 18, 1, 17, 2, 19, 16, 11, 22]. Compared with these previous work, MetaStream focuses on the problem of content discovery in peer-to-peer "caching and relaying", and addresses the server load, network cost, and client heterogeneity issues in a practical and efficient framework.

7 Conclusions

This paper presents the design and implementation of MetaStream, a scalable and distributed content discovery protocol for high bandwidth and low network cost media streaming in a cooperative environment. Specifically, this paper makes the following contributions:

- We presented the design and evaluation of MetaStream, which creates a media distribution network based on a topology-aware overlay and allows streaming clients to achieve high bandwidth in caching-and-relaying on-demand media while incurring low link cost to the underlying network;
- We conducted a large-scale evaluation of 1000 streaming clients running in an emulated 5050-router network topology. The experiment results show that MetaStream incurs low network link stress and low server load by effectively locating nearby client-side streaming sources, and the streaming overlay is highly resilient to overlay membership changes and client failures.

Acknowledgment

This work was supported in part by an NSF CAREER award (ACI-0238379).

References

1. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, August 2002.
2. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Content Distribution in Cooperative Environments. In *Proceedings of ACM SOSP*, October 2003.
3. Y. Chae, K. Guo, M. M. Buddhikot, S. Suri, and E. Zegura. Silo, Tokens, and Rainbow: Schemes for Fault Tolerant Stream Caching. *IEEE JSAC, Special Issue on Internet Proxies*, April 2002.
4. S. Chen, B. Shen, S. Wee, and X. Zhang. Adaptive and Lazy Segmentation Based Proxy Caching for Streaming Media Delivery. In *Proceedings of ACM NOSSDAV*, June 2003.
5. Y. Cui, B. Li, and K. Nahrstedt. oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. *IEEE JSAC, Special Issue on Recent Advances in Service Overlays*, 2004.
6. Y. Cui and K. Nahrstedt. Layered Peer-to-Peer Streaming. In *Proceedings of ACM NOSSDAV*, June 2003.
7. A. Dan and D. Sitaram. A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads. In *Proceedings of SPIE MMCN*, January 1996.
8. D. Eager, M. Vernon, and J. Zahorjan. Minimizing Bandwidth Requirements for On-Demand Data Delivery. *IEEE Transactions on Knowledge and Data Engineering*, 13(5), 2001.
9. L. Gao and D. Towsley. Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast. In *Proceedings of IEEE ICMCS*, 1999.
10. V. K. Goyal. Multiple Description Coding: Compression Meets the Network. *IEEE Signal Processing Magazine*, 2001.
11. M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. Promise: Peer-to-Peer Media Streaming Using Collectcast. In *Proceedings of ACM Multimedia*, November 2003.

12. K. Hua. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems. In *Proceedings of ACM SIGCOMM*, September 1997.
13. Y. hua Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM SIGMETRICS*, June 2000.
14. S. Jin and A. Bestavros. Cache-and-Relay Streaming Media Delivery for Asynchronous Clients. In *Proceedings of NGC*, October 2002.
15. S. S. Kien A. Hua, Ying Cai. Patching: A Multicast Technique for True Video-on-Demand Services. In *Proceedings of ACM Multimedia*, September 1998.
16. D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of ACM SOSP*, October 2003.
17. Z. Li and P. Mohapatra. HostCast: A New Overlay Multicasting Protocol. In *Proceedings of IEEE ICC*, May 2003.
18. J. Liebeherr, M. Nahas, and W. Si. Application-Layer Multicasting with Delaunay Triangulation Overlays. In *Proceedings of IEEE GLOBALCOM*, May 2001.
19. V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient Peer-to-Peer Streaming. In *Proceedings of IEEE ICNP*, November 2003.
20. S. Ramesh, I. Rhee, and K. Guo. Multicast with Cache (Mcache): An Adaptive Zero-Delay Video-on-Demand Service. In *Proceedings of IEEE INFOCOM*, April 2001.
21. R. Rejaie, M. Handley, and D. Estrin. Layered Quality Adaptation for Internet Video Streaming. *IEEE JSAC, Special Issue on Internet QOS*, 2000.
22. R. Rejaie and A. Ortega. PALS: Peer-to-Peer Adaptive Layered Streaming. In *Proceedings of ACM NOSSDAV*, June 2003.
23. S. Sheu, K. A. Hua, and W. Tavanapong. Chaining: A Generalized Batching Technique for Video-On-Demand Systems. In *Proceedings of IEEE ICMCS*, June 1997.
24. A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of USENIX OSDI*, December 2002.
25. E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE INFOCOM*, March 1996.

The Scalability of Swarming Peer-to-Peer Content Delivery

Daniel Stutzbach[1], Daniel Zappala[2], and Reza Rejaie[3]

[1] University of Oregon, Eugene, Oregon
{agthorr, reza}@cs.uoregon.edu
[2] Brigham Young University, Provo, Utah
zappala@cs.byu.edu
[3] University of Oregon, Eugene, Oregon

Abstract. Most web sites are unable to serve content to a large number of users due to the inherent limitations of client-server file transfer. Recent peer-to-peer content delivery protocols have demonstrated the feasibility of spreading this load among the clients themselves, giving small web sites the possibility of serving large audiences with very low cost. In this paper we use a simulation-based performance evaluation to study the fundamental question of the scalability of swarming peer-to-peer content delivery. Our results demonstrate the superior scalability of swarming with respect to load, file size, block size, and client bandwidth.

1 Introduction

While the Web has transformed the way we use the Internet, it can still be very slow when large numbers of users try to access a single web site at the same time. Access can be particularly slow when users are trying to search a database or download a large file (such as a software update). Today's state-of-the-art in this area is a Content Distribution Network (CDN), which replicates a web server's content at various locations in the network, then provides a mechanism to redirect users to a local replica.

Although a CDN is feasible for large content providers who expect high access rates, the cost is often not justifiable for what we call "ordinary users" – organizations with unpredictable demand or smaller budgets, such as schools, governments, small companies, and individuals with personal web sites. Several alternatives are available to these users, but none are satisfactory. Buying more bandwidth is certainly one possibility, but most users are limited in the amount they can pay and cannot afford to always provision for peak demand. More commonly, users can recruit volunteers to setup a mirror of the web site so that load is spread among several servers. Ultimately, there are far too many "ordinary users" compared to the pool of volunteers willing to help. Another potential solution is proxy caching, but this is useful primarily from the perspective of an individual client for whom the cache is available. From the perspective of the web server, caching must be deployed at a wide number of sites in order to be

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 15–26, 2005.

effective at reducing load. One last alternative is to multicast content from the web server to a group of clients [1], but multicast is not widely deployed, requires loose synchronization among clients, and additional mechanisms to accommodate heterogeneous client bandwidths.

It is becoming increasingly clear that the best way for this majority of "ordinary users" to serve a large audience is through the use of peer-to-peer technology. A new type of data transfer called *swarming* leverages the cooperative nature of peer-to-peer networking to serve large numbers of users without placing a heavy burden on a centralized web server. With swarming, any user that has downloaded some piece of content from a server can then itself act as a server to other peers for that content. Because no single peer has the entire content, nor a high amount of bandwidth, peers download content from each other in parallel [2, 3], constructing the larger file from the pieces they collect. This frees the web server from having to deliver the entire file to all users; instead it gives a piece of the file to some users and then relies on those users to exchange data among themselves. This technology is most commonly used by BitTorrent clients [4], though other variations have also been proposed [5].

In this paper, we focus on the fundamental question of the scalability of swarming peer-to-peer content delivery. While existing systems such as BitTorrent indicate the feasibility of swarming in general, the load observed on these systems (for a single file) is still small enough that their scalability have not yet been demonstrated.[1] Accordingly, we have designed a simple swarming protocol and examined its scalability through a comprehensive, simulation-based performance evaluation. Our study demonstrates for the first time that peer-to-peer downloading can scale with offered load far beyond what client-server file transfers can deliver. We also show that swarming can smoothly handle flash crowds, with minimal effect on client performance. Finally, we demonstrate that swarming can scale to a wide range of file sizes, block sizes, and client bandwidths.

2 Simple Swarming Protocol

In order to study swarming performance, we have designed a simple swarming protocol. We believe this protocol generalizes the basic content delivery mechanism used by existing peer-to-peer software such as Gnutella, BitTorrent, and Slurpie [5]. Our design divides peer-to-peer content delivery into four key components: *swarming initiation*, *peer identification*, *peer selection*, and *parallel download.* While there are many design choices for each component, our goal is to use a simple yet effective design for each component. This enables us to study the performance of swarming delivery while minimizing complex dynamics and interactions among the components of the system. This makes it easier to correlate an observed behavior to a particular mechanism or parameter.

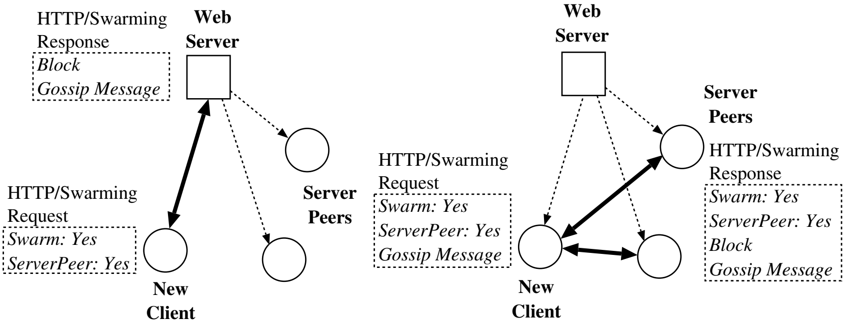
Our design also integrates the swarming protocol into a standard web server. The system uses client-server communication to bootstrap peer location and to

[1] One study [6] measured 51,000 peers in 5 days, but this is only 7 peers per minute.

serve as a fallback in case a client's known peers all leave the network. The system uses peer-to-peer networking to deliver content and to discover additional peers through gossiping [7, 8]. Swarming is implemented on top of HTTP, providing backward compatibility and allowing for incremental deployment. It is important to note that the client may be either a web browser or proxy server. It should be simple to integrate swarming into existing proxies because they already include server functionality. We also note that swarming protocols can be used in a wide variety of content delivery systems, and are not limited to the web.

2.1 Protocol Overview

Swarming clients send regular HTTP requests to web servers, along with two additional headers. The Swarm header indicates that a client is willing to use swarming, and the Server Peer header indicates that a client is willing to provide the requested file to other peers (Figure 1(a)). Web servers that are not capable of swarming will simply ignore the unrecognized headers.



(a) A new client gets a block and a gossip message.

(b) The client finds and downloads blocks in parallel.

Fig. 1. Protocol Example

To initiate swarming, a web server gives clients a single *block* of the file and a *Gossip Message* (Figure 1(a)). A block is a portion of the file, typically tens or hundreds of kilobytes. The server determines the block size on a per-file basis; our performance evaluation looks at a range of file and block sizes. A gossip message contains a list of peers that are willing to serve portions of the same file. For each peer, the message lists the peer's IP address, a list of blocks the peer is known to have, and a time stamp indicating the freshness of this information.

When a client receives a swarming response (partial content plus a gossip message), it invokes a *peer selection* strategy to determine the subset of peers from which it will download content. A client's primary concern is to locate blocks of the file that it has not yet received. The client then begins downloading blocks from both the web server and the peers in parallel. In this study we are not concerned with fairness; thus, unlike BitTorrent, we do not include incentives to encourage clients to upload data.

Each transaction between a client and the web server or a peer includes a single block and a two-way exchange of gossip messages (Figure 1(b)). Requesting a single block at a time will naturally lead to faster peers delivering more blocks, resulting in proportional load balancing. Exchanging gossip messages with peers enables progressive *peer identification* – clients gradually learn about other peers in the system. This is needed because the initial pool of peers identified by the web server may refuse to serve the client, may disconnect from the network, may have low bandwidth connectivity, or may simply not have all of the content the client needs. Gossiping provides a low-overhead mechanism for peer identification while distributing this load away from the web server and among all peers.

When a peer receives a request for a block, it determines whether it will accept the connection based on its configuration or capabilities. The peer then delivers the requested block and exchanges gossip messages with the client. Once the peer has itself downloaded the entire file, it may decide to leave the system immediately or it may choose to linger and help additional peers. In our study we use a *lingering time* of zero in order to model the case where users are selfish and automatically exit after completion of the download. Existing studies on BitTorrent assume that clients linger for a longer time; one measurement study found that approximately 40% of the file was provided by clients who had already downloaded the entire file [6]. We do not want to assume that such charitable behavior will continue. When a peer disconnects from the system, it does not wait for any ongoing block downloads to finish. To reduce the amount of bookkeeping that is required, clients discard partially downloaded blocks.

As large numbers of clients attempt to download the same content, they form a dynamic mesh or swarm of peers. We can view this mesh as a *collaborative delivery system*, where peers with larger portions of the file or higher bandwidth will tend to serve greater numbers of clients.

2.2 Protocol Algorithms and Parameters

Our swarming protocol uses the following algorithms and parameters:

Swarming Initiation. We have chosen the conservative approach of swarming at all times. This enables the web server to be proactive with regard to load, so that it doesn't react too late to a flash crowd.

Peer Identification. Our goal for this component is to discover recent peers, since peers may leave the system at any time. Each client caches a record for the N_c peers with the most recent time stamp, then includes in its gossip messages the most recent N_g peers, where $N_g \leq N_c$. Clients must also gossip about peers that are no longer available, to minimize time wasted trying to connect to these peers. Accordingly, disconnected peers are represented as peers without any blocks of the file, and these records are shared in gossip transactions like any other. Eventually the record of a disconnected peer is discarded because its time stamp will never be renewed.

Peer Selection. We use a simple strategy that emphasizes content availability. Each client limits itself to N_d concurrent downloads. When choosing a new peer, clients choose the peer that has the most blocks that it still needs.

Parallel Download. Each client chooses N_d peers for parallel download, using the peer selection component, then continues to use this set unless a peer disconnects or runs out of blocks that the client needs. In either of these cases, the client drops the peer and then immediately invokes the peer selection component to choose a replacement. If none of the peers in the client's gossip cache have blocks that the client needs, then the client contacts the web server for some additional peers. When requesting blocks, a client selects a block randomly from those available when connecting to both the web server and peers. This ensures some amount of diversity in the content that is available, and increases the chance that a client can find a peer with content that it needs.

3 Performance Evaluation

We have conducted an extensive performance evaluation of swarming using our own peer-to-peer simulator built on top of some of the original *ns 1.4* code. We summarize our results here; for more details see our technical report [9]. Unless otherwise mentioned, we use the default swarming parameters given in Table 1.

Similar to congestion control studies, we use a simplified topology in which we model the Internet as a single router, as shown in Figure 2. This topology helps us to focus on the places where bottlenecks are likely to occur under high load – at the web server and peers. In order to focus on transmission delay, we set the propagation delay of all links to 1 *ms*. Most of our simulations use the basic scenario shown in Table 2, which models a server at a small company and clients with broadband access. We use a 1 MB file for most simulations because this provides a good comparison with a standard web server. Swarming can be used for much larger files, but a larger file in this case would render the web server completely useless.

We control the workload by varying the arrival rate of clients requesting the same file from the web server. For a given arrival rate, we randomly generate

Table 1. Default Swarming Parameters

Parameter	Value
N_d (Concurrent downloads)	4
N_c (Size of gossip cache)	64
N_g (Peers in gossip message)	10
Block Size	32 KB

Table 2. Basic Swarming Scenario

Parameter	Value
File size	1 Megabyte
Server bandwidth	$1 Mbps$
Client bandwidth (down/up)	$1536 Kbps$ / $128 Kbps$

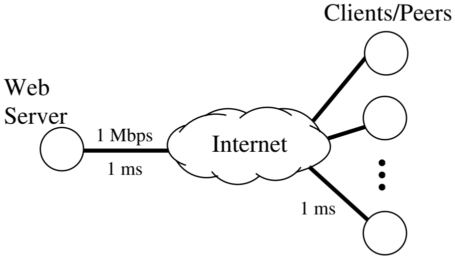


Fig. 2. Simulation Topology

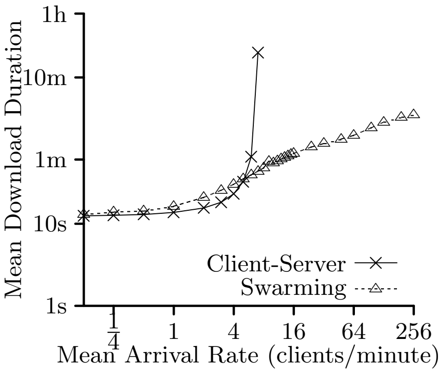


Fig. 3. Scalability with Load

client inter-arrival times using an exponential distribution. We also simulate a flash crowd by abruptly increasing the arrival rate for a given period of time.

When a client arrival occurs, we create a new client and it immediately begins its download. During the download, the client also serves content to other peers, then it leaves the system once its download is complete. While in the real world clients may be somewhat more polite, we opt for a conservative approach and hence underestimate the benefits of swarming.

Our primary performance metric is client download time. In particular, we are interested in how download time changes in response to increased load, rather than the absolute value of download time. We also measure the packet loss rate, the number of clients served by each peer, the number of blocks served by each peer, and a variety of other swarming-related metrics. Unless otherwise indicated, we begin each simulation with a warm-up period of 500 download completions, allowing the system to reach steady state behavior. We then collect data for 5500 download completions. For each experiment we conduct multiple runs of our simulations, average the results, and compute the 95% confidence interval. We do not include confidence intervals here because they are very small.

3.1 Scalability: Load

Swarming has excellent scalability with respect to load. In Figure 3, we plot the mean time a client takes to fully download a file versus the client arrival rate on a log-log scale. Client-server is unable to handle load beyond about 7 clients per minute; at this rate the arrival rate exceeds the departure rate. Naturally, this point will vary, depending on server bandwidth, file size, and load. Swarming, on the other hand, can serve at least 192 clients per minute, which translates to serving the one megabyte file to more than a quarter million people per day. This is an impressive feat for a $1 Mbps$ access link. To serve an equivalent load using a client-server protocol would require, at a bare minimum, $28 Mbps$. This could cost thousands, perhaps tens of thousands, of dollars per month!

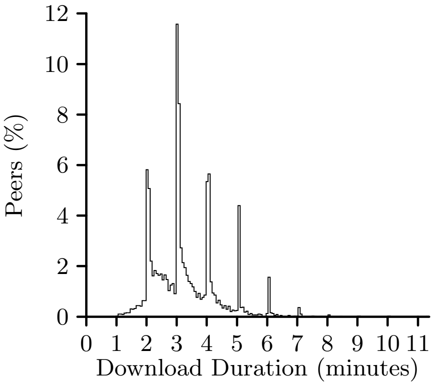


Fig. 4. Download times: 192 clients/min

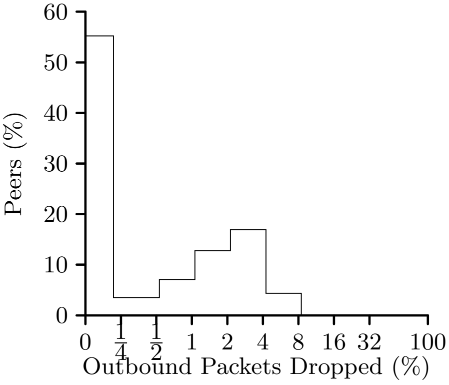


Fig. 5. Packet loss: 192 clients/min

Due to memory limitations we were unable to simulate higher loads, but a quick calculation suggests that any bound on swarming performance will not occur for at least an order of magnitude further increase in arrival rate.[2] This limitation exists for any scheme that relies on contacting a known, central point to initiate a download. At extremely high loads, swarming can incorporate a decentralized method for locating peers, such as PROOFS [10].

We observed several performance limitations with our swarming protocol, each of which could be fixed with further optimizations. First, our protocol imposes a small performance penalty under light load because the web server delivers all blocks, but with the added overhead of gossip messages. This could be eliminated by designing a dynamic server initiation component that uses swarming only when needed. Second, under very high load – 192 clients per minute – the server experiences severe packet loss. This is due to the web server providing at least one block to every client. This could be relieved by having the server only give a subset of the clients a block and require the rest to get the entire file from their peers. Finally, under high load a disproportionate number of download times are close to multiples of 60 seconds (Figure 4). It is important to note that the download times are measured individually from the start of each client; thus, this pattern does not indicate synchronization of flows within the network. This behavior is caused by our use of a 60 second time out to detect dead connections, and the high congestion at the server causes many clients to time out. This suggests that alleviating server congestion will also result in significant improvements for the peers.

Our swarming protocol otherwise performs very well. Even at the highest load we simulated, peers experience very little packet loss (Figure 5). Furthermore, during high load the burden of content delivery is spread evenly among the peers. At the highest load we simulated, roughly 60% of the clients serve less than one

[2] We assume a single 1500-byte packet is used to transmit the referral information. The $1Mbps$ server can transmit $2^{20}/(1500 * 8)$ of these per second, or 5242 per minute.

megabyte. Nearly all of the clients upload less than two megabytes. Re-serving the file once or twice is fair, so this behavior is quite good.

3.2 Scalability: Flash Crowds

While good steady-state behavior is important, web servers must also be able to cope with extreme bursts of activity called flash crowds. We simulate the effect of a flash crowd by abruptly increasing the arrival rate for a fixed period of time. The steady state load in this section is 6 clients per minute. For client-server transfer we introduce an impulse of 12 clients per minute, lasting for one hour. For swarming, we provide a more challenging flash crowd by increasing the flash crowd rate to 120 clients per minute![3] In both cases, after the flash crowd passes, we simulate the steady state load until the web server is able to recover. The results are presented in Figure 6 and Figure 7, where each data point represents the mean download time for all downloads finishing in the previous 1000 seconds.

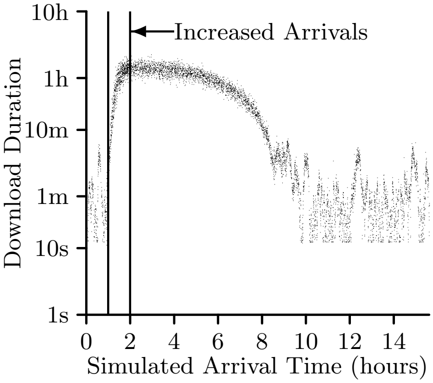


Fig. 6. Client-Server flash crowd

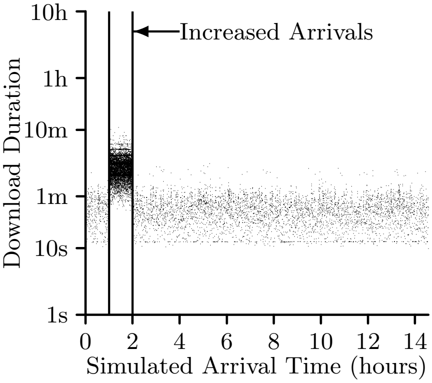


Fig. 7. Swarming flash crowd

As can be seen from these figures, swarming enables a web server to smoothly handle large flash crowds that would otherwise bring content delivery to a crawl. It maintains reasonable response times as the crowd arrives, and dissipates the crowd quickly. With the traditional client-server approach, the arrival rate of the clients quickly exceeds the service rate, and the server does not recover until long after the arrival rate decreases. Note that we intentionally do not model clients giving up during a long download period so that our model for the web server matches that of swarming. This is not unrealistic, as Kazaa users have been observed to wait for a day to download MP3 files [11].

It is particularly impressive that we achieve this result using a conservative and unoptimized swarming protocol. With additional optimizations, the server

[3] In order to fill out the graph, we ran this simulation for 12,000 completions.

should be able to handle even larger flash crowds. In fact, by utilizing Gnutella or PROOFS [10] to locate peers under extremely high loads, *swarming can be made effectively immune to flash crowds.*

3.3 Scalability: File and Block Size

Swarming also scales well with the size of the file, allowing a small user to easily serve large files (e.g. multimedia). We demonstrate this result in Figure 8, which shows the mean download time for both swarming and client-server as a function of the file size. For this simulation we use an arrival rate of 4 clients per minute, with the same basic scenario given in Table 2. Varying the file size is similar to varying the arrival rate in that both cases increase the load on the web server. Swarming exhibits only a linear increase in download time as the size grows; note that a linear increase is the best it can do because the file size is growing while the bandwidth at the client stays constant. For file sizes of two megabytes or larger the client-server protocol is unable to enter steady-state.

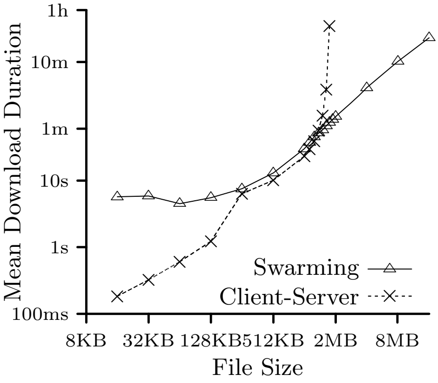


Fig. 8. Scalability with file size

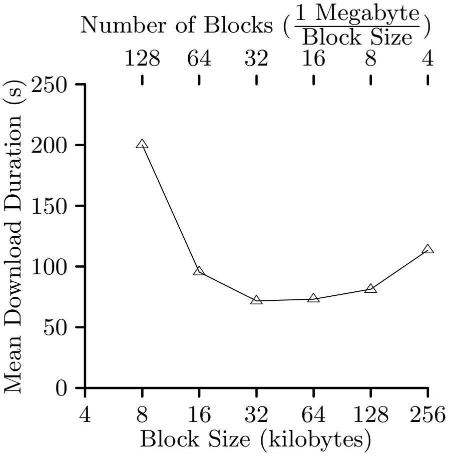


Fig. 9. Scalability with block size

An interesting result from this simulation is that gossiping can impose significant overhead when the block size is small. For this simulation the number of blocks is 32, regardless of file size. Thus as the file size decreases, the gossip message becomes large relative to the size of the data. This is shown in Figure 8, in the region where file size is less than 256 KB; the mean download time never goes below 4 seconds. Despite this overhead, swarming will eventually outperform client-server for small files as the arrival rate increases. Nevertheless, this is clear evidence that swarming can benefit from dynamic server initiation.

To fully explore the effect of block size on swarming performance we conducted a series of simulations with varying block and file sizes, using an arrival rate of 16 clients per minute. Figure 9 shows the results of one these simulations, using a file size of 1 megabyte, from which we can identify two trends.

First, download time increases as the block size decreases. As the block size becomes smaller, the transmission delay incurred by the client transmitting a gossip message becomes a significant part of the overall delay. Second, as the block size increases the download time increases slightly for large blocks. This is a result of the "last block problem", which occurs when the last block to be downloaded is coming from a slow source. Our results show that for a block size of 256 KB the last block consumes 35% of the download time. BitTorrent solves this problem by simultaneously downloading the last block from multiple sources, although this results in redundant data transmission

3.4 Scalability: Client Bandwidths

Finally, swarming can handle a wide range of client bandwidths. This is an important result since some peer-to-peer protocols collapse when too many low-bandwidth users enter the system; the original Gnutella protocol had this flaw. To address this concern, we conducted a variety of simulations using different mixtures of clients drawn from three classes: modem ($56Kbps/33Kbps$), broadband ($1536Kbps/128Kbps$), and office ($43Mbps$). We assign each class of users a different probability, then randomly assign new clients to one of these classes according to these probabilities. Other than client bandwidth, the rest of the scenario is the same as Table 2.

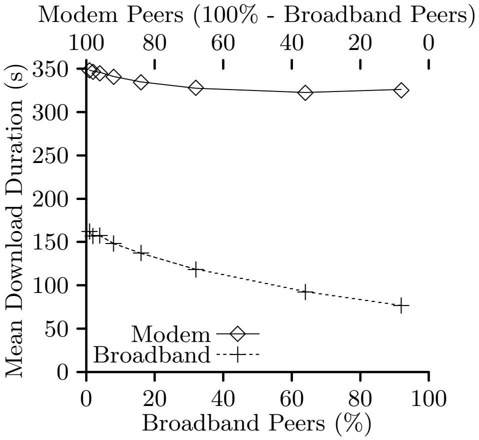


Fig. 10. Impact of low-bandwidth clients

As an example of our results, Figure 10 plots the mean download time for a combination of broadband and modem users. As the percentage of modem users increases from 10% to 99%, the download time for broadband users increases by roughly a factor of two. While this is a significant increase, the system clearly continues to function well despite an overwhelming number of low-bandwidth users. We see a similar result for office users – their mean download time increases by a factor of 3 for the same changes in the mix of broadband and modem users. Note that broadband users do not get a significant benefit from office users because office users do not linger after downloading the file and because we are not using any kind of bandwidth-based peer selection. Finally, the performance of modem users is relatively unchanged by large numbers of higher-speed users because their access link remains a bottleneck.

4 Related Systems

A number of peer-to-peer content delivery systems have been developed recently that use the concept of swarming. BitTorrent [4] is notable as a swarming system because it is currently used to transfer large files, such as new software releases, to hundreds of peers. BitTorrent uses a centralized host for peer identification and includes mechanisms to try to enforce fairness among peers. Slurpie [5] forms a content delivery mesh, with the main goal of trying to minimize client download time. Slurpie clients utilize bandwidth estimation to adapt to changing conditions in the mesh and select peers for data transfer. With both CoopNet [12] and Pseudoserving [13], a web server gives clients a list of possible peers, and the client chooses a single peer from which it downloads the entire content.

Several other peer-to-peer content-delivery protocols take different approaches. Splitstream [14] divides content into several stripes and then multicasts each stripe to a different application-layer multicast tree. Peers join as many trees as they want, and nodes try to spread the load of content delivery among themselves. Bullet [15] organizes peers into a mesh and delivers disjoint sets of data to peers. Peers are individually responsible for discovering and retrieving data.

Other peer-to-peer systems focus on enabling web servers to cope with high load. The Backslash system [16] forms a collaborative network of web mirrors. An overloaded web server then redirects clients to a cached copy of the content located at one of the collaborating sites. PROOFS [10] uses a peer-to-peer network of clients to cache popular content. When a client is unable to download content from a web server, it queries the peer-to-peer network to see if any other user has a copy of the desired content.

5 Conclusions and Future Work

The power of swarming as a file transfer mechanism is that it actually uses scale to its advantage – system capacity increases with the number of peers participating. Peers spread the load of content delivery over the entire network and share the burden of peer identification with the web server; this prevents server overload and avoids network congestion. Clients utilize parallel download to protect themselves against peer instability, which would otherwise hinder a peer-to-peer application. Moreover, swarming is clearly an economical solution for the web server because it does not have to pay for the bandwidth used for peer-to-peer communication. Clients have an incentive to participate in swarming because the alternative is that everyone suffers from a congested server.

Our study lays the groundwork for future research in many interesting areas. A dynamic server initiation component should be able to decide when to use client-server transfer (for small unpopular files) and when to use swarming (for large or popular files). It should also be able to switch exclusively to swarming during high loads. Other avenues of research include applying swarming to dynamic content, bandwidth-based and distance-based peer selection, dynamic adjustment of the number of concurrent downloads, peer performance monitoring, more efficient gossiping, and ways to encourage cooperation.

References

1. Clark, R.J., Ammar, M.H.: Providing Scalable Web Services Using Multicast Communication. Computer Networks and ISDN Systems **29** (1997) 841–858
2. Byers, J., Luby, M., Mitzenmacher, M.: Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In: IEEE INFOCOM. (1999)
3. Rodriguez, P., Biersack, E.W.: Dynamic Parallel-Access to Replicated Content in the Internet. IEEE/ACM Transactions on Networking (2002)
4. Bram Cohen: Bit Torrent (2004) http://bittorrent.com.
5. Sherwood, R., Braud, R., Bhattacharjee, B.: Slurpie: A Cooperative Bulk Data Transfer Protocol. In: INFOCOM. (2004)
6. Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P.A., Hamra, A.A., Garces-Erice, L.: Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In: Passive and Active Measurement Workshop. (2004)
7. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: ACM Symposium on Principles of Distributed Computing. (1987) 1–12
8. Karp, R.M., Schindelhauer, C., Shenker, S., Vocking, B.: Randomized rumor spreading. In: IEEE Symp. on Foundations of Computer Science. (2000) 565–574
9. Stutzbach, D., Zappala, D., Rejai, R.: Swarming: Scalable Content Delivery for the Masses. Technical Report UO-TR-2004-01, University of Oregon (2004)
10. Stavrou, A., Rubenstein, D., Sahu, S.: A Lightweight, Robust P2P System to Handle Flash Crowds. In: IEEE ICNP. (2002)
11. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In: ACM Symposium on Operating Systems Principles. (2003)
12. Padmanabhan, V.N., Sripanidkulchai, K.: The Case for Cooperative Networking. In: 1st International Workshop on Peer-to-Peer Systems. (2002)
13. Kong, K., Ghosal, D.: Mitigating Server-Side Congestion on the Internet Through Pseudo-Serving. IEEE/ACM Transactions on Networking (1999)
14. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: SplitStream: High-Bandwidth Content Distribution in a Cooperative Environment. In: International Workshop on Peer-to-Peer Systems. (2003)
15. Kostic, D., Rodriguez, A., Albrecht, J., Vahdat, A.: Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In: SOSP. (2003)
16. Stading, T., Maniatis, P., Baker, M.: Peer-to-Peer Caching Schemes to Address Flash Crowds. In: 1st International Workshop on Peer-to-Peer Systems. (2002)

Leopard: A Locality Aware Peer-to-Peer System with No Hot Spot[*]

Yinzhe Yu, Sanghwan Lee, and Zhi-Li Zhang

Department of Computer Science and Engineering,
University of Minnesota,
Minneapolis MN 55455, USA
{yyu, sanghwan, zhzhang}@cs.umn.edu

Abstract. Recent research [7, 12, 2] has shown that Internet hosts can be efficiently (i.e., without excessive measurements) mapped to a virtual (Euclidean) coordinate system, where the geometric distance between any two nodes in this virtual space approximates their real IP network distance (latency). Based on this result, in this paper, we propose an alternative approach that *inherently* incorporates a virtual coordinate system into a P2P network. In our system, called Leopard, a node is assigned a coordinate in the so-called *node geo space* as it joins the network, and obtains neighbor relationships that reflects network proximity from the beginning. The object id space and the node geo space are then "weaved" together via a novel technique called *geographically-scoped hashing.* Through analysis and simulation, we show three major desirable properties of Leopard to exemplify the power of this paradigm shift: i) a constant routing stretch, i.e., IP level network latency of object look-up is proportional to the distance between a requesting node and the target object; ii) always locates a near-by copy when multiple copies exist; and iii) effectively handles "flash crowd" traffic with near optimal load balancing.

Keywords: peer-to-peer, lookup service, virtual coordinates, locality-awareness.

1 Introduction

A fundamental challenge in Peer-To-Peer (P2P) systems is how to locate objects of interest in the network, namely, the *look-up service.* A key break-through towards a scalable and distributed solution of the lookup problem is the *distributed hash table (DHT)* (including Chord[9], CAN[11], Tapestry[14] and Pastry[10], among others). However, since both object (any entities of interest) id and network node are *randomly hashed* to a same id space, "locality-awareness" is *not inherent* in the basic design of DHT. As a result, *routing stretch* of object look-up, defined by the ratio of the network distance traveled by a look-up query message

[*] This work was supported in part by NSF grant ITR-0085824 and CNS-0435444.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 27–39, 2005.

and the distance between the requesting node and the nearest copy of the target object, can be high. Another challenging problem is to effectively cope with the so-called "flash crowd" or "hot spot" problem, namely, a sudden surge of user requests for a popular object. Since in standard DHT an object is randomly hashed to a single id (or a few id's, if replicated using several hash functions) in the id space, the node(s) responsible for storing or answering queries for that object can be overwhelmed by a "flash crowd," creating a "hot spot" in the system.

Recent research [7, 12, 2] has shown that Internet hosts can be efficiently (i.e., without excessive measurements) mapped to a virtual (Euclidean) coordinate system, such that the geometric distance between any two nodes in the virtual space accurately approximates their real IP network distance (latency). For example, [7] shows that with an 8-dimension virtual space, more than 90% of inter-nodal distances in the virtual space are within 50% error margin of their real network distances. One of the intended usage of such a virtual coordinates system is to improve the neighbor relationship in the P2P overlay. For example, a node can gradually select close-by fingers (routing neighbors) in the virtual space to reduce routing stretch. However, such *incremental* change to existing scheme may not be the best way to maximize the utility of this new facility.

In this paper, we propose to inherently incorporate locality-awareness into a P2P system. We separate the *object id space* from the *node space*: each object is assigned a unique id in an object id space; while each node is assigned a *coordinate* in a so-called *(node) geo space*, as it joins the network. During the join process, the node obtains neighbor relationships that reflects "network proximity" *from the beginning*. The object id space and the node geo space are then "weaved" together via a technique called *geographically-scoped hashing* (GSH): each object id is mapped into multiple points (i.e., coordinates) in the node geo space with varying geographical scopes; the nodes that are closest to these points are responsible for maintaining "pointers" to the object and performing look-up queries for it. Through analysis and simulations, we demonstrate that: i) Leopard has a constant IP-level routing stretch; this holds not only in an *average* or *probabilistic* sense, but also in the *worst-case*. ii) Leopard always locates a *near-by* copy when multiple copies exist. iii) Leopard effectively handle "flash crowd" with near optimal load balancing, without adding *exogenous* complexity in look-up operations.

In the remainder of the paper, we first present an overview of our scheme in Section 2. The design details of Leopard are described in Section 3 and 4, focusing on object operations and node space management respectively. Performance evaluation results based on packet level simulations are included as Section 5. Finally, we briefly compare our work with related works in Section 6 before concluding in Section 7.

2 Key Concepts: Area Hierarchy and GSH

Same as standard DHT, we assign each object a unique *identifier* from an *id space*, based on either application semantics or random hashing. Without loss

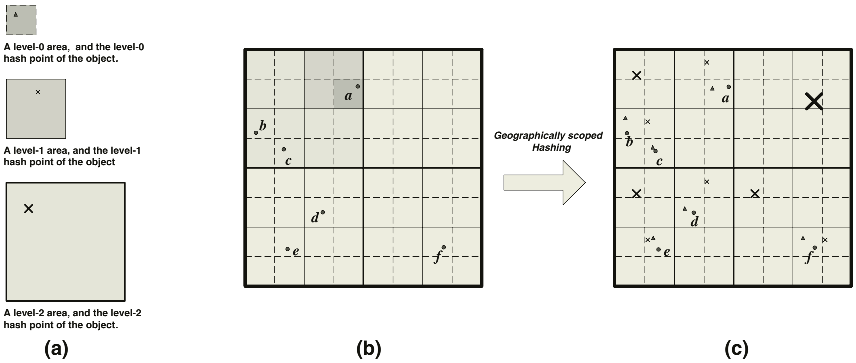

A level-0 area, and the level-0 hash point of the object.
A level-1 area, and the level-1 hash point of the object
A level-2 area, and the level-2 hash point of the object.
(a)
a
b
c
d
e
f
(b)
Geographically scoped Hashing
(c)

Fig. 1. An example of geographically-scoped hashing. (a) An object is hashed to different relative points in different levels of areas. (b) Locations of six owners $(a-f)$ of a same object ω. Note that the level-$0,1,2$ areas of node a are highlighted with different shades. (c) Hash points (pointer nodes) of the six owners at various levels of area

of generality, we assume that the object id is a scalar (e.g., a 128-bit number). We use $\omega.ID$ to denote the identifier of an object ω. On the other hand, nodes form the *node geo space* – a finite d-dimensional metric space – upon which a coordinate system is defined. A nodes's *coordinate* is determined when it joins the system. The coordinate can be the node's actual geographical location obtained via GPS, or a virtual coordinate obtained via a virtual coordinates service such as GNP [7], Virtual Landmark [12] or Vivaldi [2]. For simplicity of exposition, we assume that the node geo space is a d-dimensional Euclidean space, with inter-nodal distance approximating IP level network distance.

We introduce a *hierarchical grid* over the node geo space by dividing it into a hierarchy of (d-dimensional) *areas*: at the highest level, the entire space is the level-L area, where L is a system parameter specifying the total number of levels in the hierarchy. For $1 \leq l \leq L$, each level-l area is divided into 2^d level-$(l-1)$ areas, obtained by cutting the level-l area into half along each of the d dimensions. Fig.1(b) illustrates such a hierarchy of areas with $L = 3, d = 2$, where square areas of different levels are delineated with different line styles. For $l = 0, 1, \ldots, L$, let A_l be the level-l area containing a node a. In Fig. 1(b), we use four different levels of shades to show $A_0 \ldots A_3$. We see that $A_l \subset A_{l+1}, l = 0, ..., L-1$, and A_L is the entire node geo space. Let r_l denote the *size* of a level-l area A_l (i.e., its side length), then $r_{l+1} = 2r_l$. We will use $\mathcal{O}(A_l)$ to denote A_l's *origin* coordinate, i.e., point in A_l with the smallest coordinates.

We now introduce the concept of *geographically scoped hash* functions. For $l = 0, 1, \ldots, L$, let $\mathcal{H}_l$ be a d-dimensional random hash function[1] with the range $[0, r_l)^d$. Given an object ω and a level-l area A_l, the *hash point* of ω under (the geographical scope) A_l is a point within A_l given as $\mathcal{H}(\omega, A_l) = \mathcal{O}(A_l) +$

[1] A d-dimensional hash function can be constructed as a Cartesian product of d independently 1-dimensional random hash functions.

$\mathcal{H}_l(\omega.ID)$. In Fig. 1(a), we illustrate the hash functions for three areas (scopes) of different levels. Fig. 1(b) shows the locations of six nodes $(a - f)$, each owning a copy of the object ω. Fig. 1(c) shows the hash points of the six nodes in various areas. The node in the node geo space that is *closest*[2] to the hash point $\mathcal{H}(\omega, A_l)$ of object ω is referred to as the *level-l pointer node* for object ω in area A_l, and is denoted by $\mathcal{P}(\omega, A_l)$. Pointer nodes of an object are responsible for maintaining object "pointers" and answering look-up queries.

We now give a high-level illustration on how these hash points are used for locating objects in Leopard. When a node a wishes to share object ω, it *publishes* ω by "planting" a pointer at pointer node $\mathcal{P}(\omega, A_0)$. For $l = 1, \ldots, L-1$, each level-l pointer node $\mathcal{P}(\omega, A_l)$ in turn computes the next-level hash point $\mathcal{H}(\omega, A_{l+1})$ and plants a pointer at $\mathcal{P}(\omega, A_{l+1})$. Now suppose a node b (let B_l denote its level-l areas) is interested in ω. It first sends a look-up query to $\mathcal{P}(\omega, B_0)$. Note that if a and b reside in the same level-0 area, then $\mathcal{P}(\omega, B_0)$ (=$\mathcal{P}(\omega, A_0)$) will be able to direct node b to node a for the object. Otherwise, $\mathcal{P}(\omega, B_0)$ computes the level-1 hash point and forwards the query to $\mathcal{P}(\omega, B_1)$. The process goes on recursively. If nodes a and b reside in the same level-l area, i.e., $A_l = B_l$, then the level-l pointer node $\mathcal{P}(\omega, B_l)$ $(= \mathcal{P}(\omega, A_l))$ will have a pointer to object ω, and thus can direct node b to node a for object ω. As $A_L = B_L$, in at most L steps, node b will be able to locate a pointer to object ω. In a sense the pointer nodes of an object ω form a distributed search tree (embedded in the node geo space), where each edge connects $\mathcal{P}(\omega, A_l)$ to $\mathcal{P}(\omega, A_{l+1})$, as shown in Fig. 2(a).

3 Leopard Look-Up Service

In Leopard *pointers* – information about objects – are stored in various pointer nodes. To reduce the storage cost in pointer nodes and to shield high level pointer nodes from frequent publish/withdraw operations in the large area it is in charge, *Leopard only maintains precise information (the IP address) of individual object owners at level-0 pointer nodes*. Each level-0 pointer node of an object ω has an *owner list* that lists all the object owners of ω in this level-0 area. In higher-level pointer nodes Leopard maintains *aggregate* information: a `TRUE/FALSE` value (called a *branch indicator*) for each of its 2^d next lower level areas, indicating whether that area contains at least one object owner. By following a series of branch indicators with `TRUE` values down a branch of the object search tree the IP address of an object owner can be obtained at the level-0 pointer node (leaf of the tree).

Fig. 2(a)(left) depicts an object search tree with $d = 2$ and $L = 3$, for five object owners $a - e$, located in positions as shown in Fig. 1. A black node represents an *active* pointer node, a pointer node currently storing a pointer to the object. A white node represents an *inactive* pointer node — a node closest to a hash point in the area, but stores no pointer because there is no object owner

[2] We will give a precise definition of *closeness* in Section 4.

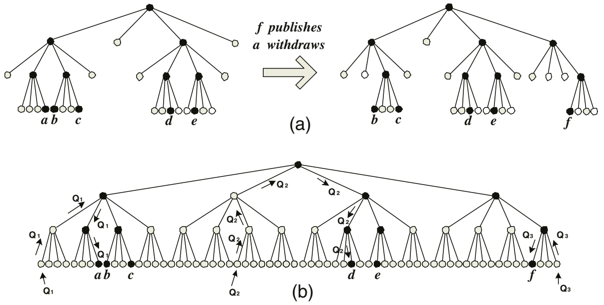


Fig. 2. (a) Initially, a object search tree (left) have five owners (a, b, c, d, and e). After f publishes and a withdraws, the object search tree becomes the one on the right. (b) Three examples (Q_1, Q_2, Q_3) of query message path

in that area yet. Each active pointer node has $2^d = 4$ children. An active child corresponds to a `TRUE` branch indicator and an inactive child corresponds to a `FALSE` branch indicator. Note that not all the inactive pointer nodes are shown in Fig. 2(a) (e.g., children of an inactive pointer node).

To publish an object ω, an owner a in a level-0 area A_0 first computes the hash point $\mathcal{H}(\omega, A_0)$, and sends a publish request to its level-0 pointer node $n := \mathcal{P}(\omega, A_0)$[3]. If n is currently *active*, i.e., it already has a pointer for ω, it simply appends node a's IP address and coordinate to the object owner list in the existing pointer. Otherwise, it creates a new level-0 pointer, and notifies the level-1 pointer node $\mathcal{P}(\omega, A_1)$ that its area now contains an owner of ω. This process continues recursively until either an active pointer node or the root (top-level) pointer node is reached. During the process, the corresponding branch indicators at pointer nodes in every levels are set accordingly. Algorithm 1 list the steps of a publish operation. It can be viewed as a recursive RPC(remote procedure call). When a new owner a want to publish an object ω, it simply calls $n.\mathbf{publish}(\omega, 0, a.coord, a.IP)$, where $n := \mathcal{P}(\omega, A_0)$.

The object withdraw operation involves the similar recursive process to adjust the object owner list and the branch indicators in pointers of various level of pointer nodes. Figure 2(a) shows an example where node f publishes and node a withdraws. The example shows that by storing the aggregate information of "whether a branch contains an owner of ω or not," high-level pointer nodes are often not affected when nodes publish and withdraw.

When a node x in a level-0 area X_0 wants to look up for an object ω, it sends a query for object ω to the pointer node $\mathcal{P}(\omega, X_0)$. If the pointer node contains a level-0 pointer for ω, it replies to x with the IP address of an owner. Otherwise, the pointer node recursively queries the higher-level pointer nodes, until a pointer to object ω is found. Starting at that pointer node, the query is sent down to

[3] As we will show in Section 4, by including a target point in the packet header and performing greedy forwarding, a packet can be sent from any source node towards any point (e.g., a hash point) in the node geo space.

Algorithm 1 : n.**publish**$(\omega, l, a.coord, a.IP)$ //obj,level,coordinate,IP addr

```
//lookup the pointer database for an entry with the three-field key
$entry \Leftarrow$ lookupPointer$(\omega, l, \mathcal{H}(\omega, A_l))$
if $l = 0$ then
  //always create a new pointer and append it to owner list
  storePointer$(\omega, l, \mathcal{H}(\omega, A_l), a.coord, a.IP)$
  if $entry =$ NULL then
    //publish at higher level recursively
    $n' \Leftarrow \mathcal{H}(\omega, A_{l+1})$
    $n'$.publish$(\omega, l+1, a.coord, n.IP)$
else
  if $entry =$ NULL then
    //create a new pointer only if it's the first owner in the area
    $entry \Leftarrow$ storePointer$(\omega, l, \mathcal{H}(\omega, A_l))$
    $n' \Leftarrow \mathcal{H}(\omega, A_{l+1})$
    $n'$.publish$(\omega, l+1, a.coord, n.IP)$
  //determine branch indicator index using level l and a's coordinate
  $br\_id \Leftarrow$ getBranchID$(l-1, a.coord)$
  //set branch indicator accordingly
  $entry.branch[br\_id] \Leftarrow$ TRUE
return
```

the lower-level pointer nodes by following a branch where the branch indicators are TRUE[4], until a level-0 pointer node is reached. The level-0 pointer node then replies node x with the IP address of an owner. Due to space limitation, we refer the reader to the technical report version of this paper [13] for a detailed list of Leopard Query algorithm. Figure 2(b) illustrates three examples (Q_1, Q_2 and Q_3) of object query paths.

As shown in the example, when an owner a and a querying node x reside in a common level-l area A_l, the query can be performed in $2l$ (logical) steps in the tree. In fact, the distance traveled by a query message is also bounded by $O(r_l)$, where r_l is the size (side length) of a level-l area, as stated in the following theorem[5]:

Theorem 1. *Suppose a node x queries for an object ω in Leopard, and the located object owner shares a level-l area with x. The total geometric distance traveled by the query message (summing up all the $2l$ steps) in the node geo space is bounded in worst case by $4\sqrt{d}r_l$, where r_l is the size of a level-l area.*

Note that Theorem 1 gives a *worst-case* bound on the geometric distance traveled by a query message from the requester to the *located owner*. Leopard can also be optimized (using what we call "sibling indicators" as described in [13]) so that the

[4] When there are multiple TRUE branch indicators, the pointer node can either choose one of them randomly, or employ certain scheduling strategy, such as round robin.

[5] Due to space limit, we refer [13] for all proof of theorems in this paper.

located owner is near-optimal (by a constant factor) compared with the optimal owner (closest to the requester). In particular, we have the following theorem.

Theorem 2. *Suppose a node x is querying for an object ω. Let the owner located by Leopard be s_1, and the closest owner in the network be s_2. Let $D(a, b)$ denote the distance between two points a and b in the geo space. Then we have either $D(x, s_1) - D(x, s_2) \leq 2\sqrt{d}r_0$ or $\frac{D(x,s_1)}{D(x,s_2)} \leq 4\sqrt{d}$.*

We note that that from Theorems 1 and 2, the distance traveled by a query message to locate an object in Leopard is at most a constant factor of the optimal distance, i.e., the geometric distance between the requester and the closest object owner in the node geo space. If we assume that the distance in the node geo space accurately approximates IP level network distance, then Leopard achieves a constant routing stretch at IP level even in the worst case.

3.1 Mitigating Hot Spots

To cope with the "flash crowd" problem, Leopard imposes the following simple rule on nodes requesting for an object:*A node x starting the transfer(downloading) of an object ω from another node (located through Leopard) must publish ω for at least the duration of the object transfer; in addition, x must honor any transfer request accepted during its publishing period.* This rule creates an object propagation model similar to the popular Bit-Torrent system. However, since Leopard always returns a nearby copy of an object, it not only releases the "hot spot" on the hosting nodes (as also achieved by Bit-Torrent), but also greatly reduces the routing cost of the "flash crowd", since each request is resolved "locally." The object search tree embedded in the node geo space achieves good load balance naturally without maintaining a complex object tracker, as Bit-Torrent does.

We next show that based on this rule, Leopard effectively mitigates the "flash crowd" problem with excellent load balancing. We measure the balance of load with two metrics. The first metric, *owner service count*, is defined as the number of object transfer requests an object owner serves during its publishing time (assuming it withdraws immediately after finishing downloading itself). Ideally, this metric can be as low as 1 even with extremely high query rate, i.e., each requesting node can serve the next requester. The second metric, *pointer node service count*, is the number of query messages a pointer node of ω serves during its downloading period of ω. We have the following theorem regarding the upper bound of the two metrics.

Theorem 3. *i) The number of object transfer requests an object owner a serves during its publishing period is bounded by $L + n_0$, where n_0 is the number of nodes in the level-0 area A_0. ii) The number of queries a pointer node a serves during* one downloading period *is bounded by $L + n_0$ (for level-0 pointer node) or $L + 2^d$ (otherwise).*

Since we have $L = O(\frac{\log{(N/n_0)}}{d})$ (N being total node number), the two metrics are bounded by $O(\frac{\log N}{d})$ and $O(\frac{\log N}{d} + 2^d)$ respectively. Therefore, Theorem 3 guarantees that *regardless of the object request rate*, the two metrics grow proportional to $\frac{\log N}{d}$.

4 Node Space Management

In Leopard each node a is not only assigned a d-dimensional coordinate, (denoted by $(x_1^a, \ldots, x_d^a)$), but is also responsible for a portion of the node space around its coordinate called a *zone*. The node geo space is divided into zones that satisfy the following two properties: 1) the boundaries of a zone are hyperplanes perpendicular to axis of dimensions; and 2) a node's zone always contain its coordinate. A node a's zone, denoted by Z_a, can be represented by d ranges $\{[u_1^a, v_1^a), [u_2^a, v_2^a), \ldots, [u_d^a, v_d^a)\}$, such that $u_i^a \leq x_i^a < v_i^a, \forall 1 \leq i \leq d$. A *neighbor* zone of Z_a is a zone adjacent to it on a boundary. Formally, Z_b is a neighboring zone of Z_a if $\exists i \in \{1, \ldots, d\}$ such that $u_i^a = v_i^b$ or $v_i^a = u_i^b$, and $\forall j \in \{1, \ldots, d\} \setminus \{i\}, [u_j^a, v_j^a) \cap [u_j^b, v_j^b) \neq \Phi$, i.e., there exists a dimension where the two zones are adjacent, for the remaining dimensions, the two zones overlap. Each node maintains a *neighbor table* about neighbor zones: the ranges of neighbor zones and the coordinate and IP address of the nodes owning them. Zones are dynamically created and re-structured as nodes join and leave the node geo space. Note also that the zone structure is independent of the area hierarchy defined earlier.

4.1 Greedy Forwarding in Node Space

As we mentioned earlier, the actual packet forwarding in Leopard is based on a greedy algorithm guided by a destination coordinate stamped in the packet header. To forward a packet to a destination point p in node geo space, every intermediate node simply forward it to the neighbor with a zone *closest* to p, until the current node is already closest (than any neighbor), in which case the current node is the destination. To measure the *closeness* between a zone and a point in the node geo space, we define the distance $\mathcal{D}(Z_a, p) = \min\{d_q | \forall q \in Z_a, d_q = ||q - p||\}$, i.e., the distance between the closest point in the zone (to p) and the point p. The rationale behind this definition of distance is two folded: 1) it's easy to calculate[6]; 2) based on this definition, our greedy forwarding algorithm ensures delivery of packet, as guaranteed by Theorem 4. Theorem 4 rules out the possibility of the *local minima* problem associated with many greedy algorithm, and guarantees delivery of packet in finite number of steps.

Theorem 4. *With the above definition of distance between a zone Z_a and a point p in the node space, a node always has a neighbor closer to p, unless it is the destination node ($p \in Z_a$).*

[6] Let q^* denote the closest point in Z_a (to p), we can obtain its coordinates as follows.

$$x_i^{q^*} = \begin{cases} x_i^p \text{ if } x_i^p \in [u_i^a, v_i^a) \\ u_i^a \text{ if } x_i^p < u_i^a \\ v_i^a \text{ if } x_i^p \geq v_i^a \end{cases}$$

Therefore, $\mathcal{D}(Z_a, p) = \mathcal{D}(q^*, p)$. Fig. 3(a) shows the corresponding closest point q^*'s from three different points p_1, p_2 and p_3 to a rectangular zone.

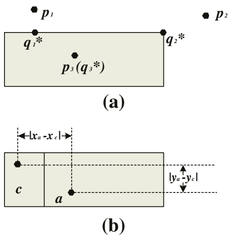


Fig. 3. (a) Determine closest point in a zone to a target point. (b) Zone splitting example

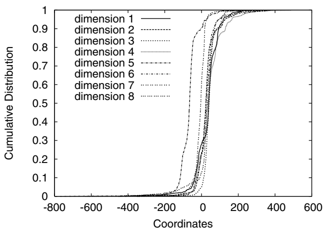


Fig. 4. Cumulative distribution of coordinates in a GNP data set

4.2 Node Join and Leave

When a node a wants to join Leopard, it follows the procedure below.

1. a obtains its coordinates through an external service such as GNP.
2. a finds a *bootstrap* node b in Leopard, through a well-known rendezvous point (e.g., a URL).
3. a sends out a `Join Request` message through the bootstrap node b, carrying a's coordinate as the destination. Using the greedy forwarding algorithm, the `Join Request` message will arrive at the node currently owning a zone that contains a's coordinate, denoted by c.
4. a and c split the zone. To do so, they first choose a dimension in which a and c's coordinates have the largest difference. They then divide the zone into two parts at the mid-point along the chosen dimension. Each part serves as one node's new zone. Figure 3(b) shows an example of zone splitting.
5. After obtaining new zones, a and c each updates its neighbor table. Note that both a and c's new neighbor sets are subsets of node c's original neighbor set.

When a node a leaves the network gracefully, it performs the following procedure.

1. a withdraws all objects it is currently publishing.
2. a notifies its neighbors of its intention to withdraw. Those neighbors with zones *mergable* with a's zone will take over, appropriately dividing a's zone into sub-zones if necessary, so that they can be merged with their own zones. (The sub-division of the zone is necessary to maintain the zone properties postulated at the beginning of this section.)
3. a transfers all the state information (pointers) to those nodes that are taking portion of its zone.

5 Simulation Experiments

We vary three key parameters in our packet level simulation experiments: number of dimensions d, number of levels L, and the node distribution in geo space. We

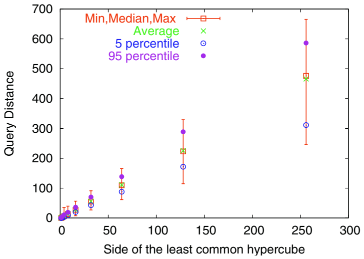


Fig. 5. Query distance vs. smallest common area size. d=2, L=8, uniform

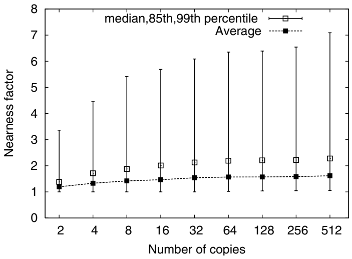


Fig. 6. Nearness factor vs. number of object copies. $d = 2, L = 8$, uniform

use $d = 2$ and 8 as $d = 2$ reflects the actual geographical coordinates; while $d = 8$ is shown [7, 12] to be enough for accurate approximation. The hierarchy level L is determined such that the number of nodes in level-0 areas does not exceed a small constant. In addition to the simple uniform node distribution, we generate node distributions that model realistic Internet nodes to address the concern of pointer node load balancing. This is done by using the coordinate distributions of the GNP data sets [1]. Fig. 4 shows the coordinate distribution on each of the 8 dimensions for the 869 virtual coordinates generated by GNP.

Query Distance. To verify Leopard *routing stretch*, we define *query distance*, computed by summing up the geometric distances in node geo space of each forwarding hop of a query message. Provided that the virtual coordinate system is accurate, query distance is a reliable indicator of real network distance. We construct a network of 10^5 uniformly distributed nodes, and distribute $1,000$ objects into random nodes such that the i^{th} $(1 \leq i \leq 1000)$ object has i copies. We generate $100,000$ queries from random node for a random target object, and record the query distances for every query. Fig. 5 shows the distribution of the query distances grouped by the *smallest common areas* of the querying nodes and the *located* owner. The x-axis in Fig. 5 represents the size (i.e., side length) of the smallest common area, normalized by r_0. We observe a clear linear relation between the query distance and the common area size, which strongly support our analysis results about Leopard's constant routing stretch. In addition, the actual value of that "constant stretch" is small: less than 2 for average, and less then 2.5 for 95 percentile.

Locating Nearby Copies. We first define a metric called *nearness factor*, which is the ratio of the distance from the querying node to the *located* owner and that distance to the actual *nearest* owner in the network. We construct a system $(d = 2, L = 8)$ with 10^5 uniformly distributed nodes and 100 unique objects, each with 2^k copies in the network (we vary the value of k from 1 to 9 in nine different experiments). For each experiment, we generate 5000 queries from random nodes for a random object. Fig. 6 shows the statistics of nearness factors. The average, median, 85th percentile and 99th percentile are computed for the nine experiments with different k's. We see that the nearness factor is small: the *medians* are 1 for all k values, indicating at least half of the querying nodes being able to find the actual *nearest* copy.

Flash Crowd Mitigation. We generate queries to a *single* popular object with different query rates and measure the object owner service count (OSC) and pointer-node service count (PSC). The simulation starts with a 10^5 node network, and a randomly chosen node publishes the target object. Random queries are then generated with a Poisson distribution with a mean of 2^q queries per second ($q = 0 \dots 6$). We performs five runs of the experiment. Each run lasts $1,000$ seconds, i.e., for $q = 6$ per second, $64,000$ nodes obtain a copy. Since we fix object downloading time at 100 seconds, and node always withdraws immediately after downloading, the system maintains 100×2^q copies in steady state. Two different sets of system parameters are used: i) $d = 2, L = 8$, uniform node distribution and ii) $d = 8, L = 13$ with non-uniform distribution. Table 1 shows the histograms (accumulated over five runs) of nodes with different OSCs. For example, when query rate is 1/s (uniform), a total of 4991 ($2654 + 2 \times 948 + 3 \times 133 + 4 \times 8 + 5 \times 2$) requests are served in five runs. Only two nodes ever served five requests. For all query rates, vast majority of nodes (99.5%+ and 99.9%+ in two distributions) serve three requests or less. Based on Table 1, we plot the fraction of nodes has OSCs of 1 through 5, as shown in Fig.7. We observe that the fractions are almost identical, indicating Leopard's capability to achieve near-optimal load balancing regardless of query rate. Finally, we note that we also observe similar pattern in our analysis of PSC [13].

Table 1. Histogram of owner service counts at different query rates. Left: uniform distribution; Right: non-uniform distribution

rate\count	1	2	3	4	5	6	7	8	9	11	12	13	15	21	25	28	rate\count	1	2	3	4	5	6	7	9	10	11	12	19
1	2654	948	133	8	2	0	0	0	0	0	0	0	0	0	0	0	1	3160	830	64	2	0	0	0	0	0	0	0	0
4	10762	3847	472	48	5	3	2	0	0	0	0	0	0	0	0	0	4	12869	3144	205	16	5	0	0	0	0	0	0	0
16	42635	15419	1734	177	29	3	5	1	2	1	0	0	1	0	0	0	16	53232	12203	679	59	7	3	0	0	1	0	0	0
64	165901	62293	8192	902	138	29	10	0	1	1	2	1	1	2	1	1	64	216818	47483	2278	160	13	5	1	1	2	1	1	1

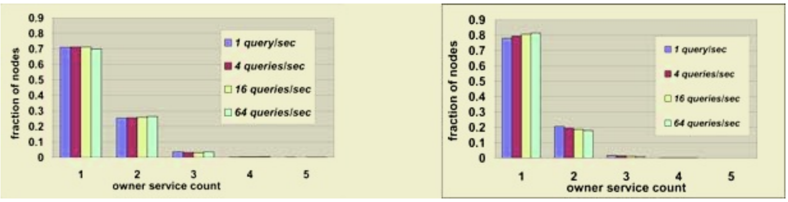


Fig. 7. Fraction of nodes that has various owner service counts. Left: Uniform dist.; Right: Non-Uniform dist

6 Related Works

Tapestry [14] and Pastry [10] are early DHT systems that can locate nearby object. However, since they use randomly hash node id, the "nearby" copy located is in the sense of the node virtual space.More recent efforts of reducing routing stretch and locating nearby object include Coral [3], Canon [4]and BeeHive [8]. Coral divides the P2P network into several levels of self-similar network, each level employs an original chord network. As Coral relies on a mechanism to gradually cluster nodes and split/merge clusters as the network evolves, its effectiveness hinges largely on the efficiency of the cluster algorithm. BeeHive advocates proactive caching in the P2P network to alleviate hot spot problem, relating the amount caching to the query rate. However, in the real world, it may be difficult to decide the query rate a priori, especially in the sudden surge of a flash crowd. The main contribution of this paper is to demonstrate the feasibility of inherently incorporating locality-awareness into P2P overlay. Finally, we note that the hierarchical area structure adopted in Leopard has also been used in other routing/loop-up schemes. For example, GLS [5] uses it in mobile node location lookup. However, as we have shown, P2P look-up service has many unique problems, such as managing multiple copies of an object, constructing a node overlay that preserves neighbor proximity. Leopard is also superficially similar to DIM [6], proposed for sensor networks, with similar terminologies like "zone," "sibling node," etc. In DIM, node neighbor relationships are formed based on geographical closeness (rather than random hashing). However, DIM is mainly designed for range queries, and always stores one copy of an object/event, while Leopard aims to find the nearest copy of an object in the network.

7 Conclusion and Future Work

We have proposed to incorporate locality-awareness inherently into the P2P network and have demonstrated many desirable properties of Leopard. Our future plan includes detailed comparisons between Leopard and Pastry/Tapestry on IP level stretch and capability of finding nearby copy, Leopard operation issues such as dynamically determining level number, and their efficient implementations.

References

1. Global Nework Positioning. http://www-2.cs.cmu.edu/ eugeneng/research/gnp/.
2. F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of ACM SIGCOMM*, 2004.
3. M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *Proc. of USENIX NDIS*, 2004.
4. P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in g major: Designing dhts with hierarchical structure. In *Proc. of IEEE ICDCS*, 2004.
5. J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proc. of MobiCom*, Aug. 2000.

6. X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 63–75. ACM Press, 2003.
7. T. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proc. of IEEE INFOCOM*, June 2002.
8. V. Ramasubramanian and E. G. Sirer. Beehive: Design and implementation of next generation name service for the internet. In *Proc. of ACM SIGCOMM*, 2004.
9. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proc. of ACM SIGCOMM*, Aug. 2001.
10. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale p2p systems. In *Proc. of IFIP/ACM Middleware*, 2001.
11. I. Stoica, R. Morris, D. Karger, M. Kaashock, and H. Balakrishman. Chord: A scalable P2P lookup protocol for Internet applications. In *Proc. of ACM SIGCOMM*, 2001.
12. L. Tang and M. Crovella. Virtual landmarks for the Internet. In *Proc. of ACM IMC*, Oct. 2003.
13. Y. Yu, S. Lee, and Z.-L. Zhang. Leopard: A locality-aware peer-to-peer system with no hot spot, Tech. Report CSE Dept., U of Minnesota, 2004.
14. B. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE J-SAC*, 22(1), 2004.

PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications

David Hausheer[1] and Burkhard Stiller[2, 1]

[1] Computer Engineering and Networks Laboratory TIK,
Swiss Federal Institute of Technology, ETH Zurich, Switzerland

[2] Department of Informatics IFI, University of Zurich, Switzerland
{hausheer, stiller}@tik.ee.ethz.ch

Abstract. P2P-based applications like file-sharing or distributed storage benefit from the scalability and performance of completely decentralized P2P infrastructures. However, existing P2P infrastructures like Chord or Pastry are vulnerable against selfish and malicious behavior and provide currently little support for commercial applications. There is a need for reliable mechanisms that enable the commercial use of P2P technology, while maintaining favorable scalability properties. PeerMint is a completely decentralized and secure accounting scheme which facilitates market-based management of P2P applications. The scheme applies a structured P2P overlay network to keep accounting information in an efficient and reliable way. Session mediation peers are used to minimize the impact of collusion among peers. A prototype has been implemented as part of a modular Accounting and Charging system to show PeerMint's practical applicability. Experiments were performed to provide evidence of the scheme's scalability and reliability.

1 Introduction

Emerging peer-to-peer (P2P) applications benefit from the large amount of resources provided by many individual peers. Using sophisticated techniques for aggregation and replication of these resources, P2P-based systems are able to provide a much higher robustness and performance than traditional client/server-based applications. For example, file-sharing applications like eMule [8] or BitTorrent [5] are able to provide access to huge amount of content in a reliable way. At the same time, an increasing number of applications make use of basic P2P network infrastructures like Chord [20] or Pastry [17] and benefit from the good scalability properties of these systems.

However, many existing P2P infrastructures and applications suffer from peers which behave in a selfish or malicious fashion [9], [18]. Moreover, there is currently little support for commercial applications for which appropriate accounting and charging mechanisms are required. Compared to centralized systems, accounting of resource usage is much more complex in a distributed environment and misusage of such mechanisms is more difficult to prevent.

PeerMint is a decentralized accounting scheme that provides the ability to keep track of contribution and consumption of resources by peers. The scheme uses remote

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 40 – 52, 2005.

peers to store and aggregate accounting information in a trustworthy and scalable way. The aggregated accounting information can be used to enforce fair sharing of resources between peers or as a basis for additional charging and payment mechanisms. A structured P2P overlay network is applied to map accounts onto a redundant set of peers and organize them in an efficient and scalable manner. Other than similar work (cf. [1], [15], [16], [22]), the proposed scheme uses session mediation peers to maintain session information about transactions between peers. This minimizes the impact of colluding peers trying to increase their account balance without actually contributing resources. Additionally, PeerMint provides economic flexibility by supporting the use of different types of tariffs. The proposed scheme is secure in that it ensures the availability and integrity of the accounting data. However, it does not provide confidentiality or privacy, as every peer is, in principle, able to access the accounting data of any other peer.

PeerMint has been designed and implemented as an accounting mechanism within a generic and modular Accounting and Charging (A&C) system for P2P applications [11]. The system separates generic accounting functionality such as session maintenance and account configuration from individual underlying accounting mechanisms. This enables the implementation of alternative accounting schemes with different properties, independently from existing core functionality. As the interface to the accounting mechanism is very generic, the proposed scheme could also be used in other environments.

The remainder of this document is structured as follows: Section 2 presents the main principles and requirements for an accounting mechanism for P2P applications. Section 3 introduces the concept of remote accounting that is adopted in PeerMint and gives an overview on the potential design space. A detailed description about the design of PeerMint on top of Pastry is given in Section 4, while Section 5 presents its implementation within the Accounting and Charging system mentioned. In addition, the scheme is evaluated with respect to scalability and robustness. Finally, Section 6 concludes the paper and discusses some open issues.

2 Requirements and Definitions

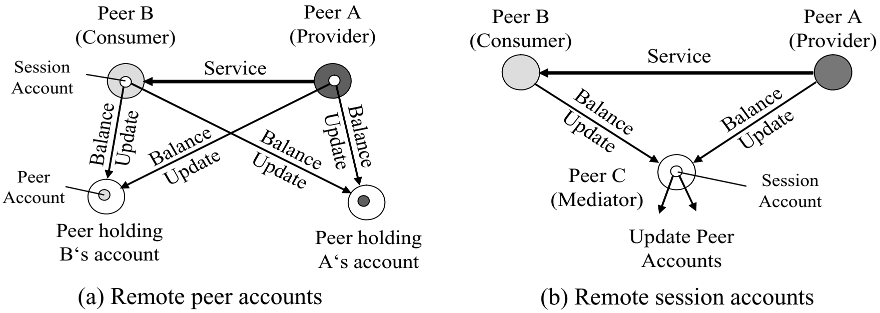
The following describes the core principles which underlie the accounting scheme proposed in this paper. The main goal of an accounting mechanisms is to ensure *accountability* [7] by providing a set of functionality that enables to account for the use of services or resources offered within a particular P2P application. As such it gives peers an incentive to contribute their own resources and serves as a basis to punish selfish behavior like free-riding. Vital accounting mechanisms are the processing of accounting events describing the amount of used resources, the application of respective tariffs, as well as the creation and maintenance of accounts to store and aggregate the accounting information and to keep track of the account balance. Accounting schemes may implement any specific type of accounting, from simple local or centralized accounting to more sophisticated remote or token-based accounting [10]. Individual accounting schemes usually fulfill specific requirements with respect to efficiency, scalability, and economic flexibility, as well as security and trustworthiness, among which there is always a trade-off. While, *e.g.*, local accounting scales perfectly with O(1), it features very bad security properties, as a peer can easily modify its account locally. Such a scheme would therefore only be suitable in trusted environments or for purposes where security is not important. Specific accounting mechanisms used in existing P2P applications are, *e.g.*, BitTorrent's tit-for-tat mechanism [5] or eMule's credit system [8]. Other related accounting mechanisms that have been developed for use in P2P systems are, *e.g.*, Karma [22], PPay [23], Mojo Nation [14] or the approach presented in [2]. Similar mechanisms have been developed for Grid computing [3], [21].

The main terms and concepts used to describe the proposed mechanism are introduced in the following. The term *session* refers to the use of a particular service or resource, *e.g.*, the download of a file or the use of some amount of computing power. A session has always two session partners, a provider peer and a consumer peer. Furthermore, *accounts* are repositories which can be used to keep and aggregate accounting information, *e.g.*, the amount of MBs uploaded or downloaded or the number of CPU cycles used. Two types of accounts are distinguished, *session accounts* and *peer accounts*. While session accounts are used to keep accounting information within a particular session, peer accounts aggregate information from several sessions, *e.g.*, the total amount of MBs uploaded and downloaded by a particular peer. Peer accounts may also be used to keep information about a peer's reputation or trustworthiness.

For every session there is a corresponding *tariff*. Its main purpose is to specify how service usage needs to be accounted for. As such it is used to process *accounting events* which are generated by the service instances running on both the provider and the consumer side of a session. A tariff is represented by a specific *tariff formula* and a set of *tariff parameters* which are previously agreed upon by both session partners. For example, a volume-based tariff could be used to account for a file download. A tariff may also contain further parameters that have to be computed dynamically during a session, *e.g.*, depending on the time of day or the balance of a particular account. Based on the result from a tariff evaluation a generic *balance update* is created and forwarded to a particular account. Note that the term balance update is used rather than *charge* to make clear that this does not necessarily imply a monetary payment. Balance updates are handled as accounting events and can thus be processed further. It depends on the applied tariff when and by how much the balance of a particular session or peer account is updated. Using tariffs, a variety of settlement schemes can be supported. For example, a peer account may be updated at the beginning or at the end of a session, which can be used to implement a pre-payment or post-payment scheme. In addition, more fine-grained updates of peer accounts are possible during a session, *e.g.*, after a particular amount of the service has been provided or a certain time has passed.

3 Remote Accounting Concept

This section introduces the basic concept of *remote accounting* as it is used by PeerMint. In general, remote accounting is based on the idea that accounts are held remotely on other peers. Remote peers are third-party peers, which are typically different from the peers currently providing or using a particular service that needs to be accounted for. Using remote accounting, accounts can be distributed and replicated over several peers, which, if designed appropriately, can increase the reliability and availability of the accounting data. In addition, a higher credibility or trustworthiness can be achieved, when many peers are involved in doing the accounting.



(a) Remote peer accounts (b) Remote session accounts

Fig. 1. Remote accounting examples

The example depicted in Figure 1(a) illustrates the basic concept of remote accounting. The scenario demonstrates two peers involved in a session. Both peers hold a local session account, while their peer accounts are held on two different remote peers. Whenever tariff evaluation results in a balance update affecting a peer account (*e.g.*, after a session was terminated), balance updates are forwarded to both peers holding the respective accounts. The remote peers collect the balance updates from both session partners. If both peers agree on the balance update (i.e. by sending equal balance updates), the peer accounts are updated accordingly. If, for any reason, the two peers disagree, the peer accounts would not be changed.

Unfortunately, the described accounting scenario faces a collusion problem. The two session partners could forward balance updates only to the peer holding the provider's account. Thus, the provider's account would be updated, while the consumer's account remained unchanged. To prevent this, peer account holders would need to interact with each other to decide whether the accounts need to be updated or not, which highly increases the accounting overhead.

The slightly different concept which underlies PeerMint is depicted in Figure 1(b). In this scenario a remote peer (mediator peer) is holding the session account for a particular service session on behalf of the respective provider and consumer peers. Both peers send their balance updates to the mediator peer which updates the session account accordingly. An ongoing session may immediately be terminated if the session partners disagree. In contrast to the previous scenario, the peer accounts which are typically held by other remote peers can only be updated by the mediator peer.

3.1 Design Space for Remote Accounting

The concept of remote accounting is very general and covers several potential subtypes. An overview on possible variants of remote accounting is given below.

Central Accounting. This is the simplest form of remote accounting and is only mentioned for completeness. Using this type of accounting, accounts would be kept on a centralized place, *e.g.*, on a large database residing on a central server. Central accounting is simple to maintain and control, and is usually highly trusted. However, the goal is to avoid any central elements in the network, as they represent a single point of failure and do not scale for a large number of peers.

Hybrid Accounting. Hybrid accounting features the simplicity of central accounting, while being more scalable with respect to the number of peers. In hybrid accounting a dedicated set of peers (so-called super-peers) are used as account holders. Super-peers are typically peers which are highly trusted by a group of peers (clients) attached to them. If the size of such a group is limited, the hybrid approach scales quite well. However, appropriate incentives need to be given to super-peers, to provide the extra accounting efforts. For instance, every peer may periodically pay a flat fee to its super-peer covering the costs for keeping and updating the accounting data.

Distributed Accounting. Distributed or decentralized accounting seems to be most promising approach for P2P applications, as it completely distributes the accounting load over all peers. Moreover, since all peers are equally involved in doing the accounting the scheme scales very well and no payments are necessary to compensate for any accounting costs. An important design dimension within distributed accounting is the redundancy of accounts. *Non-redundant* accounting describes the case, where every account is held by only one peer, while *redundant* accounting refers to accounts being replicated over several peers.

In the following, central and hybrid accounting will not further be investigated. Instead, the focus will be put on distributed accounting as adopted by PeerMint. The distributed redundant accounting case is discussed more detailed in the following section.

4 PeerMint Design

A non-redundant accounting approach as described in the previous section supersedes the need for any synchronization between accounts, however, it has some severe drawbacks. If for any reason a particular peer goes offline, accounts held by that peer would temporarily not be accessible anymore. Even worse, if a peer completely withdraws from the network, the corresponding accounting data would permanently be lost. Moreover, a malicious peer could easily modify and misreport the balance of an account it is responsible for.

Therefore, it is reasonable to introduce redundancy (cf. [19]), i.e. to replicate accounts over several peers to increase the robustness of the distributed scheme. Figure 2 illustrates the case of distributed redundant accounting as it is used in PeerMint. Both session and peer accounts are held by several independent peers. Provider and consumer peers involved in a session send their balance updates to a redundant set of m session mediation peers which are responsible for holding the session account for the current session (Phase 1). Each session mediation peer then checks if the two peers agree and updates the session account accordingly. Whenever a session account triggers a peer account update, the mediation peers send a balance update to the 2x p peers holding the respective peer accounts (Phase 2). The two phases may be repeated several times independently. To overcome byzantine failures (cf. [12]), the resulting account balance is agreed upon using majority decisions. Only if the majority of mediation peers report the same balance update, the peer accounts will be updated. Whenever a peer goes offline or permanently withdraws from the P2P network a new peer takes over its task. The new peer (shown as dashed circle) obtains the current balance from the other account holders.

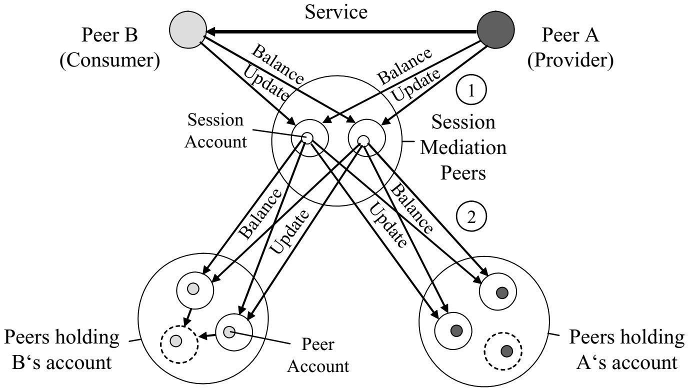


Fig. 2. Distributed redundant accounting in PeerMint

Implementing such an accounting scheme in an efficient and secure way is a complex task. Major difficulties are the mapping of accounts onto peers as well as necessary account maintenance and synchronization activities. These aspects will be addressed more detailed in the following.

4.1 Underlying Infrastructure

To map accounts onto peers, PeerMint applies a structured P2P overlay network. The scheme has been implemented on top of Pastry [17] for which an open source implementation (FreePastry) is available. However, in principle any other infrastructure (*e.g.* Chord [20]) could have been applied. This underlying infrastructure is used whenever a peer joins and leaves the network that interconnects all peers involved in PeerMint's accounting mechanism. It is assumed that all peers possess a public/private key pair which is used for peer identification and signing messages. The peers' public keys are certified by a trusted third party, which guarantees that a peer can only acquire a limited number of identities. The keys are certified offline, i.e. prior to joining the network, thus the certification process does not affect the performance of the accounting mechanism itself. An alternative method to create secure keys in a distributed way which may be adopted in future is presented in [4].

Every peer is assigned a unique 128-bit peer ID, which is calculated from the peer's public key using a secure hash function. Pastry provides an efficient prefix-based routing mechanism to find other peers in O($\log_b(N)$) hops, where N is the number of peers in the overlay network. Every node has a routing table with O($\log_b(N)$) rows which is continuously updated as peers join or leave the network. A number of n peers (called leaf-set) which are numerically closest to the current peer are part of this routing table. Similar to Karma [22], leaf-sets are used in PeerMint to map accounts onto peers as described in the following

4.2 Scheme Configuration

Every peer that participates in PeerMint's accounting mechanism (i.e. typically all peers in a particular application that uses PeerMint as underlying accounting scheme) needs to configure an instance of the scheme locally. The scheme configuration specifies the mapping function which is used to map accounts onto peers. It is important that all peers within a particular application use the same mapping function. Currently, the same hash function is applied as for calculating the peer ID, i.e. for peer accounts the hash value of the peer ID is used as key, while for session accounts a unique session key (session ID) is calculated by hashing the peer IDs of the two session partners combined with an additional timestamp. For every key there is a peer (root node) which is numerically closest to that key. The root node's leaf-set is used to hold the respective account.

Apart from the mapping rule, the peer ID of any other peer needs to be given to PeerMint which is used as bootstrap node to join an existing overlay network. Otherwise a new network is created. Also the number of redundant peers used as account holders can be configured. By default all peers in a leaf-set are used, but it is also possible to use a subset of the leaf-set or to extend the account holder set beyond the size of a leaf-set.

4.3 Account Creation and Setup

A new peer account is created when a peer joins PeerMint's accounting scheme for the first time. The peer contacts the responsible root node using Pastry's routing mechanism. Pastry routing is only used once in the beginning. All subsequent messages are directly sent over the underlying IP network. Every message exchanged between peers is signed by the sender's private key. The root node notifies all peers in the account holder set about the new peer account and sends their node handles (i.e. peer ID, IP address, and port number) back to the new peer. Peer accounts have an initial account balance, which is typically set to zero. However, a new account is created only, if none does exist so far for the same peer ID. A peer can also remove its account, if the account balance is positive. After a certain time of inactivity, a peer account is removed automatically. There is a fallback mechanism to create an account in the presence of a malicious root node. In this case, a peer tries to find another peer in the same leaf-set by recursively contacting peers on the path to the root node. This peer then temporarily takes over the role of the root node and notifies the corresponding peers. A peer can check if it is responsible for a particular account by verifying if the key of the account falls in the ID range of its own leaf-set (a different ID range applies if the account holder set is greater or smaller than the leaf-set).

Session accounts are created in a similar way. Before the session starts, the two session partners create an SLA which contains the tariff, the two peer IDs, and the session ID. The SLA is signed by both provider and consumer peer and sent to the corresponding root node responsible for the session account. The root node then forwards the SLA to all peers responsible for the new session account and again sends their node handles back to both session partners.

4.4 Account Balance Updates

Once all necessary accounts are created and set up, a session can start. An example for a session between two peers is given in Figure 3. The figure shows all peers involved in the session and the balance updates exchanged between them as described earlier. The two phases indicated correspond to those shown in Figure 2. The session partners regularly send balance updates to the corresponding session mediation peers (Phase 1).

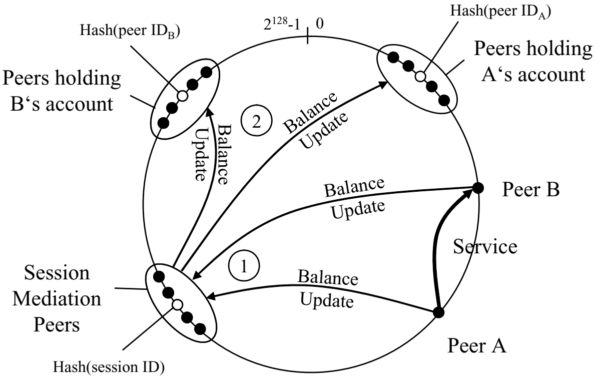


Fig. 3. Example session in PeerMint

The session mediators aggregate them and generate peer account updates as specified in the tariff. To be able to forward them to the peer accounts (Phase 2), the corresponding peer account holders are identified using Pastry routing. The signed SLA containing the session ID is used by the session mediation peers to prove that they are eligible to update the account.

4.5 Account Holder Set Maintenance

Peers may continuously join and leave PeerMint’s overlay network. Whenever a new peer joins the network, Pastry notifies the corresponding instances of PeerMint that the leaf-set has changed. Subsequently, the new peer obtains the balance of all accounts it is responsible for. Based on this information, the new peer locally creates instances of these accounts and sets their balance based on the majority of peers reporting the same balance. If no majority decision can be taken (*e.g.*, because the account is currently being updated, and therefore too many peers report different values) the new peer retries to request the account balance until a consistent value is reported. Finally, if the new peer becomes involved in an ongoing session, it notifies itself to the session partners.

Similarly, when a particular peer goes offline (i.e. does not respond anymore within a certain time), Pastry notifies the corresponding peers about the change in the leaf-set. This means that other peers will become responsible for the accounts that were held by the peer which went offline. These peers obtain the current account balance in a similar way like a new peer joining the overlay.

A peer which does not reply on synchronization requests or account queries is considered as being offline. Whenever a peer goes offline, all peers within its leaf-set send a balance update to the leaving peer's account holders to decrease its reputation. This serves as an incentive for peers to stay online and behave correctly.

4.6 Account Queries

Any peer may query the balance of a particular peer account, *e.g.*, to check whether a peer which requests a service is credit-worthy. The same procedure as described in Section 4.3 is used to find the responsible peers holding the account. In contrast to peer accounts, session accounts can only be queried by corresponding session mediation peers and the two peers involved in the session. However, there is no guarantee for the privacy of these accounts, i.e. it may be possible that an unauthorised peer is able to obtain the balance of a session account.

5 Implementation and Evaluation

The proposed scheme has been implemented in Java as part of an Accounting and Charging (A&C) system developed within the MMAPPS project [11], [13]. As such it implements a generic accounting scheme interface which is described in the following:

The first method of this interface, *configureScheme*, is used to initialize the scheme with vital information like the local peer's peer ID and further scheme-specific parameters such as the number of redundant peers used for the accounts. The method *createPeerAccount* creates a new peer account for the local peer. The same method can be used to create an account for the peer's reputation. The *createSessionAccount* method notifies the session mediation peers about a new session and hands over the corresponding SLA. Finally, *notifyBalanceUpdate* and *queryAccount* are used to update and query a particular account, respectively.

PeerMint uses the common API [6] to interact with its local instance of FreePastry, which is used as P2P communication infrastructure. As such it implements the three main methods *forward*, *deliver*, and *update*, through which a FreePastry node notifies forwarded and received messages, and changes in its leaf-set. All interactions between instances of PeerMint are completely encapsulated in Pastry messages, which are either routed over the overlay network (account holder lookup) or sent directly to the corresponding destination node (all other tasks). For each task there is a dedicated message. Based on their type, the messages are dispatched remotely and the corresponding methods are executed. Whenever a reply message is expected, *e.g.*, containing a returned account balance, a new thread is created. The corresponding thread is started by the WaitingThreadList class, when the expected message has arrived or after a certain time-out has passed. In the latter case, the corresponding message is sent again.

PeerMint has been evaluated both analytically and through experiments with the prototype in respect of overhead, scalability, and reliability (i.e. resistance against malicious or faulty peers). The overhead and scalability of PeerMint was assessed by analyzing the storage space that is needed to keep the accounting data as well as the number of messages being exchanged between peers. As mentioned in Section 4.1, there is a basic overhead of $O(\log_b(N))$ for maintaining the routing table and sending messages over Pastry (cf. [17] for a detailed analysis of Pastry's efficiency and scalability).

5.1 Analytical Evaluation

Apart from the overhead to maintain the underlying infrastructure, the following effort is needed to operate PeerMint's accounting scheme. Recall that p is the number of account holders per peer account, and m is the number of mediation peers per session account. In addition, f describes the average fraction of peers being currently online, and s is the average number of ongoing sessions. The overhead of PeerMint is shown in Table 1. As it can be seen, the overall effort is moderate and often constant in relation to N. The highest effort is needed to update peer accounts. However, these updates are usually much less frequent than session account updates. With respect to the scalability, one can see that only account holder look-ups, needed for account creation and peer account updates or queries, depend on the size of the network.

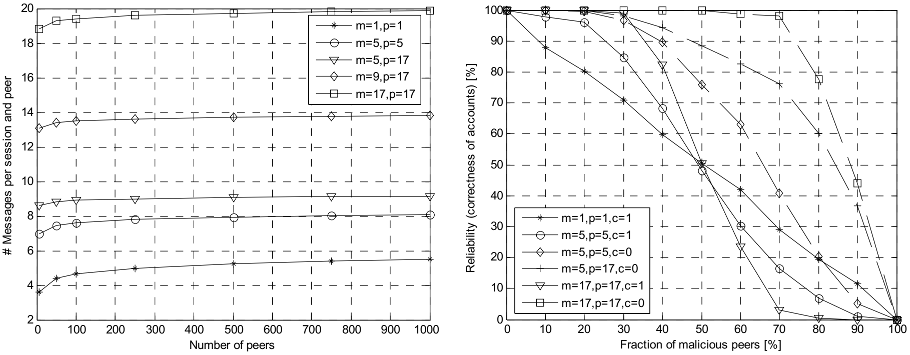
Table 1. Overhead of PeerMint

Costs	Cost driver	Peer Accounts	Session Accounts
Message costs	Account creation	$p + O(\log^b(N))$	$m + 1 + O(\log^b(N))$
	Account update	$2mp + O(\log^b(N))$	$2m$
	Account synchronization	$p - 1$	$m - 1$
	Account query	$2p + O(\log^b(N))$	$2m$
Storage costs	Avg. #accounts per peer	p / f	ms / f

Reliability denotes the ability of a scheme to perform correctly in the presence of malicious and unreliable peers. In PeerMint reliability is achieved using redundant set of peers as account holders. The size of an account holder set, p or m can be adjusted based on the fraction of malicious or faulty peers in the network. PeerMint has a statistically guaranteed reliability, if the number of account holders is higher than $3r + 1$, where r is the number of malicious or faulty nodes in an account holder set. This is the optimum that can be achieved [12].

5.2 Experimental Results

To verify and complement the analytical results, the message overhead and reliability of PeerMint has been measured in a set of simulation experiments with the implemented prototype. The experiments were run with up to 1000 peers on a testbed of four Pentium 4 CPUs, 1.8 - 2.4 GHz, with 512 MB RAM using Java VM 1.4.2. In all experiments performed the number of sessions *s* was set to 2000. For each session a consumer and provider were assigned randomly, and each account was updated once per session.



(a) Message Overhead

(b) Reliability

Fig. 4. Experimental results

Figure 4(a) shows the number of messages per session and peer for a varying number of peers *N* in the network. The size of all messages is around 1kB. It can be seen, that the messages overhead increases slowly in small networks, but levels off when the network becomes larger. Thus, the system scales very well with the size of the network.

Figure 4(b) shows the reliability of PeerMint (measured in the amount of accounts held correctly) for a selected number of account holders *m* and *p* and the fraction of malicious peers in the network. Malicious peers were modeled as account holders which reported consistent (*c*=1) or arbitrary (*c*=0) false values (*c* relates to the amount of collusion among malicious peers). As it can be seen, PeerMint can correctly keep accounts with up to 30% malicious colluding peers by using 17 peer account holders and 17 session mediators. Thus, the increase of the number of account holders results in a high reliability. Without collusion, reliability can even be achieved with as many as 50% or more malicious peers.

6 Conclusions

Accountability is the key for a success of P2P systems. Based on PeerMint, the accounting scheme presented in this paper, it is possible to provide accountability for

P2P applications in a secure and scalable manner. The paper described, how trustworthiness and resilience of accounting data can be achieved in the presence of malicious or faulty nodes, using redundant sets of independent peers. Implemented on top of an existing P2P infrastructure, PeerMint provides scalable accounting functionality at a moderate overhead. Its integration into a modular and generic Accounting and Charging system enables the flexible use of PeerMint for a variety of P2P applications. In future work, the presented accounting scheme will further be optimized and more extensively analyzed within real world environments.

Acknowledgements. This work has been performed partially in the framework of the EU IST project MMAPPS "Market Management of Peer-to-Peer Services" (IST-2001-34201), where the ETH Zürich has been funded by the Swiss Bundesministerium für Bildung und Wissenschaft BBW, Bern, under Grant No. 00.0275. Additionally, the authors would like to acknowledge discussions with all of their colleagues and project partners.

References

1. K. Aberer, Z. Despotovic: *Managing Trust in a Peer-2-Peer Information System*; Tenth International Conference on Information and Knowledge Management (CIKM'01), Atlanta, Georgia, USA, 2001.
2. Agrawal, D. Brown, A. Ojha, S. Savage: *Towards Bucking free-riders: Distributed Accounting and Settlement in Peer-to-Peer Networks*; Jacob School of Engineering Research Review, UCSD, February 2003.
3. A. Barmouta, R. Buyya: *GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration*; 17th Annual International Parallel & Distributed Processing Symposium (IPDPS 2003) Workshop on Internet Computing and E-Commerce, Nice, France, April 2003.
4. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach: *Security for structured peer-to-peer overlay networks*; Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA, USA, December 2002.
5. B. Cohen: *Incentives Build Robustness in BitTorrent*; Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.
6. F. Dabek, P. Druschel, B. Zhao, J. Kubiatowicz, I. Stoica: *Towards a Common API for Structured Peer-to-Peer Overlays*; 2nd International Workshop on Peer-to-Peer Systems (IPTP'03), Berkeley, CA, February 2003.
7. R. Dingledine, M. Freedman, D. Molnar: *Accountability*; In Peer-To-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly & Associates, Chapter 16, pp. 217 - 340, 1st edition, March 2001.
8. The eMule Project: *http://www.emule-project.net/*, October 2004.
9. P. Golle, K. Leyton-Brown, I. Mironov and M. Lillibridge: *Incentives for Sharing in Peer-to-Peer Networks*; 3rd ACM Conference on Electronic Commerce, Tampa, Florida, USA, October 2001.
10. D. Hausheer, N. Liebau, A. Mauthe, R. Steinmetz, B. Stiller: *Token-based Accounting and Distributed Pricing to Introduce Market Mechanisms in a Peer-to-Peer File Sharing Scenario*; 3rd IEEE Conference on Peer-to-Peer Computing, Linköping, Sweden, September 2003.

11. D. Hausheer, J. Gerke, B. Stiller: *A Generic and Modular Accounting and Charging System for Peer-to-Peer Applications*; 14. Fachtagung Kommunikation in Verteilten Systemen 2005 (KiVS 05), Kaiserslautern, Germany, February 2005.
12. L. Lamport, R. Shostak, M. Pease: *The Byzantine Generals Problem*; ACM Transactions on Programming Languages and Systems, Vol. 4, pp. 382-401, July 1982.
13. MMAPPS: *Market Management of Peer-to-peer Services*; EU Project, *http://www.mmapps.org/*.
14. Mojo Nation: *Technical Overview*; *http://www.mojonation.net/docs/technical_overview.shtml*, January 2002.
15. T. Ngan, D. Wallach, P. Druschel: *Enforcing fair sharing of peer-to-peer resources*; 2nd International Workshop on P2P Systems (IPTPS), Berkeley, CA, USA, February 2003.
16. N. Ntarmos, P. Triantafillou: *SeAl: Managing Accesses and Data in Peer-to-Peer Sharing Networks*; In Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland, August 2004.
17. A. Rowstron, P. Druschel: *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*. IFIP/ACM Middleware 2001, Heidelberg, Germany, November 2001.
18. J. Shneidman, D. Parkes: *Rationality and Self-Interest in Peer-to-Peer Networks*; 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA, February 2003.
19. J. Shneidman, D. Parkes: *Using Redundancy to Improve Robustness of Distributed Mechanism Implementations*; In Proceedings of 4th ACM Conference on Electronic Commerce (EC'03), San Diego, CA, USA, May 2003.
20. I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan: *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*; ACM SIGCOMM 2001, pp. 149-160, San Diego, CA, USA, August 2001.
21. W. Thigpen, T. Hacker, L. McGinnis, B. Athey: *Distributed Accounting on the Grid*; 6th Joint Conference on Information Sciences, pp. 1147-1150, 2002.
22. V. Vishnumurthy, S. Chandrakumar, E. G. Sirer: *KARMA: A Secure Economic Framework for Peer-to-Peer Resource*; Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.
23. B. Yang, H. Garcia-Molina: *PPay: Micropayments for Peer-to-Peer Systems*; ACM Conference on Computer and Communications Security (CCS '03), Washington, DC, USA, October 2003.

Highly Responsive and Efficient QoS Routing Using Pre- and On-demand Computations Along with a New Normal Measure

Yanxing Zheng[1], Turgay Korkmaz,[2] and Wenhua Dou[1]

[1] School of Computer Science, National University of Defense Technology,
ChangSha, HuNan, 410073, P.R. China
yxzheng@nudt.edu.cn
[2] Department of Computer Science,
University of Texas, San Antonio, USA
korkmaz@cs.utsa.edu

Abstract. Multi-constrained path (MCP) selection is one of the great challenges that QoS routing (QoSR) faces. To address it in an efficient and highly responsive manner, we propose a new QoSR algorithm, namely NM_MCP. Using the Dijkstra's algorithm with respect to each link metric, NM_MCP pre-computes k primary paths, where k is the number of link weights. When a routing request arrives, it executes a modified version of the Dijkstra's algorithm using a newly proposed, normal-measure-based nonlinear cost function.

Keywords: QoS routing, Pareto optimal, multi-objective optimization.

1 Introduction

One of the key issues in the next-generation networks is how to identify feasible paths that can satisfy the quality-of-service (QoS) requirements of different applications. This problem is commonly known as QoS routing (QoSR). The potential benefits of QoSR and the need for it have been recognized and acknowledged by the research community and industry [1][2]. Much work has been done on various aspects of QoSR [3][4][5]. For better responsiveness and higher success rate in identifying feasible paths, we developed a new normal measure based MCP algorithm (NM_MCP).

The rest of the paper is organized as follow. In Section 2, we formally define some related concepts. In Section 3, we review the most related studies. In Section 4, we present the proposed NM_MCP algorithm. We report simulation results in Section 5. Finally, we conclude the paper in Section 6.

2 Problem Formulation and Preliminaries

Let P_{sd} denote the set of paths between source node s and destination node d in network $G(N, E)$, where N is the set of nodes and E is the set of links.

R. Boutaba, et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 53–64, 2005.

For path $p = n_1 \rightarrow n_2 \rightarrow, \ldots, \rightarrow n_m \in P_{sd}$, each link is associated with k additive metrics. The sum of the j^{th} metric of path p can be represented as: $w_j(p) = \sum_{i=2}^{m} w_j(n_{i-1} \rightarrow n_i), j \in \{1, 2, \ldots, k\}$. Thus path p can be written as $p(w_1(p), w_2(p), \ldots, w_k(p))$.

Definition 1. *$W^k = W_1 \times W_2 \times \ldots \times W_k$ is called QoS Metric Space (QoSMS), where $w_j(p) \in W_j, j \in \{1, 2, \ldots, k\}$ for any $p \in G(N, E)$.*

Definition 2. *Mapping F is a function that maps path $p(w_1(p), w_2(p), \ldots, w_k(p))$ to a point in W^k, i.e., $F(p) = (f_1(p), f_2(p), \ldots, f_k(p)) = (w_1(p), w_2(p), \ldots, w_k(p))$.*

Definition 3. *Multi-constrained path (MCP) problem: Consider a network $G(N, E)$. Each link (u, v) is associated with a k-dimensional metric vector. Each element of w is an additive QoS metric: $w_i(u, v) \geq 0, i = 1, 2, \ldots, k$. Given routing request $c = (c_1, c_2, \ldots, c_k)$, the problem is to find a path $p \in P_{sd}$ such that $f_i(p) = w_i(p) = \sum_{(u,v)\in p} w_i(u, v) \leq c_i$, for $i = 1, 2, \ldots, k$.*

Definition 4. *Multi-constrained and Multi-optimization Problem (MCMOP): For the above MCP problem, in addition to finding a path subject to the given constraints, the objective is to*

$$\min F(p) = \min\{w_1(p), w_2(p), \ldots, w_k(p)\} \tag{1}$$

Clearly, the solutions to MCMOP will also be the solutions to MCP in Definition 3. Therefore, given a MCP problem, we can solve the derived MCMOP problem and take its solutions as that of original MCP problem. MCMOP is actually a special case of discrete MOOP that can simply be stated as follows.

Definition 5. *Multi-objective optimization problem (MOOP):*

$$\begin{matrix} \min F(x) \\ x \end{matrix} = \min[f_1(x), f_2(x), \ldots, f_k(x)] \tag{2}$$

where k is the number of objectives.

Definition 6. *Cover: Vector $u = (u_1, u_2, \ldots, u_k)$ is said to cover $v = (v_1, v_2, \ldots, v_k)$, denoted by $u \preceq v$, iff $\forall i \in \{1, 2, \ldots, k\}, u_i \leq v_i$.*

Definition 7. *Dominance: Vector $u = (u_1, u_2, \ldots, u_k)$ is said to dominate $v = (v_1, v_2, \ldots, v_k)$, denoted by $u \prec v$, iff u is partially less than v, namely $\forall i \in \{1, 2, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, 2, \ldots, k\} : u_i < v_i$.*

Definition 8. *Pareto Optimal solution: Path $p(w_1(p), w_2(p), \ldots, w_k(p)) \in P_{sd}$ is a Pareto Optimal solution iff there is no path $p'(w_1', w_2', \ldots, w_k') \in P_{sd}$ such that $(w_1', w_2', \ldots, w_k') \prec (w_1, w_2, \ldots, w_k)$*

Definition 9. *Pareto set P^* and Pareto front PF^*: For a given MOOP, $P^* = \{p(w_1, w_2, \ldots, w_k,) \in P_{sd} | p(w_1, w_2, \ldots, w_k)$ is a Pareto optimal solution of MOOP $\}$, and $PF^* = \{(w_1, w_2, \ldots, w_k) | p(w_1, w_2, \ldots, w_k) \in P^*\}$*

Elements in PF^* are also called as Pareto optimal points. If a routing request is satisfied by path $p(w_1, w_2, \ldots, w_k)$, then the request can also be satisfied by an element in PF^*. So, when we deal with MCMOP problems, we can only consider the elements in PF^*. If any Pareto optimal point cannot satisfy the routing request, the request should be refused. This property allows us to efficiently reduce the search space without compromising solutions.

3 Related Work

In general, researchers used two kinds of path length functions: linear path length function (LPLF) and nonlinear path length function (NLPLF). LPLF first takes linear combination of multiple link weights and make a single link weight for each link as $l(u \to v) = \sum_{i=1}^{k} \alpha_i w_i(u \to v)$. Combination coefficient vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is usually called as a *search direction* of Dijkstra's algorithm. Later we use Dijkstra(α) to denote the Dijkstra's algorithm that searches in direction α. Typical LPLF-based QoSR algorithms include Jaffe's algorithm [6], LARAC [7], MEFPA [8] and DWCBLA [9].

The main disadvantage of LPLF-based algorithms is that Pareto optimal points lying in the nonconvex part of Pareto front can never be found. To deal with such cases, researchers naturally considered NLPLF-based algorithms [10][11]. One main drawback of the existing QoSR algorithms using NLPLF is that they do not take the practical spread of Pareto front into consideration and thus introduce unnecessary computations in various cases [12].

In contrast to the above QoSR algorithms, we consider a new NLPLF, which is inspired by the work of Das and Dennis [13]. Das and Dennis developed a new method called *normal boundary intersection* (NBI) to obtain Pareto optimal points for an MOOP in case of *continuous* objective space. This method provides a means for obtaining an even distribution of Pareto optimal points based on the user-supplied parameters vector β, even with a nonconvex Pareto front. NBI method transforms an MOOP into the following sub-problem:

$$\text{Minimize} \quad \lambda \tag{3}$$

such that

$$\phi\beta + \lambda\hat{n} = F(x) - F^* \tag{4}$$

In this sub-problem, ϕ is a pay-off $k \times k$ matrix in which the i^{th} column is composed of the vector $F(x_i^*) - F^*$, where $F(x_i^*)$ is the vector of objective functions evaluated at the minimum of the i^{th} objective function; β is a vector of scalars such that $\sum_{i=1}^{k} \beta_i = 1$ and $\beta_i \geq 0; \hat{n} = \phi e$, where $e \in R^k$ is a column vector of ones in objective space. $\phi\beta$ is referred as the *Convex Hull of Individual Minima* (CHIM). The set of attainable objective vectors $\{F(x)\}$ is denoted by

$\hbar$. The boundary of $\hbar$ is denoted by $\partial\hbar$. NBI is in fact a technique intended to find the portion of $\hbar$ which contains the Pareto optimal points.

For a specific β, $\phi\beta$ represents a point in the CHIM. $\phi\beta + t\hat{n}, t \in R$ represents the points on normal $\hat{n}$. The solution of (3) is the point of intersection of the normal and $\partial\hbar$ closest to the origin. The constraint in (4) ensures that the point x is actually mapped by F to a point on the normal. The sub-problem given in (3) is referred as the NBI sub-problem and written as NBI_β. Different solutions would be found by varying β.

4 Proposed Algorithm

The NBI method proposed for an MOOP in case of continuous objective space cannot directly be used for MCMOP because (as clearly stated by Das and Dennis [13]) NBI may fail if the objective space is discrete as in MCMOP. Therefore, we reconsider the ideas behind NBI in the context of MCMOP problems.

4.1 Definition of Path Length

For MCMOP problems, we modify the constraint in (4) such that the points in $\partial\hbar$ can still be measured by the normal when $\partial\hbar$ is not connected. We normalize the objective function as in [14] so that the scaling deficiencies will be avoided.

Let p^{i^*} denote the path that has the minimum w_i for all paths between s and d. We can then define a *Utopia point* as

$$F^* = [f_1(p^{1^*}), f_2(p^{2^*}), \ldots, f_k(p^{k^*})]^T = [f_1^*, f_2^*, \ldots, f_k^*]^T \tag{5}$$

Let us also define a normalizing matrix as:

$$L = [l_1, l_2, \ldots, l_k]^T = F^N - F^* \tag{6}$$

where $F^N \overset{def}{=} [f_1^N, f_2^N, \ldots, f_k^N]^T$ and $f_i^N = \max[f_i(p^{1^*}), f_i(p^{2^*}), \ldots, f_i(p^{k^*})]$.

We can now define the normalized $F(p)$ as follows

$$\bar{F}(p) = [\bar{f}_1(p), \bar{f}_2(p), \ldots, \bar{f}_k(p)],$$

where $\bar{f}_i(p) = (f_i(p) - f_i^*)/l_i$.

Based on the above definitions, we define the path length as follows

$$\text{len}(p) = -\min(\lambda_i) \tag{7}$$

$$s.t. \;\; \bar{\phi}\beta + N = \bar{F}(p) \tag{8}$$

where $N = (\lambda_1 n_1, \lambda_2 n_2, \ldots, \lambda_k n_k)^T$ and $\bar{F}(p) = [\bar{f}_1(p), \bar{f}_2(p), \ldots, \bar{f}_k(p)]^T$.

The meaning of $\hat{n} = (n_1, n_2, \ldots, n_k)^T$ is the same as the one in (4) except that it is now pointing away from the origin. By denoting $\hat{\bar{\phi}}\beta$ by a vector $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)^T$, we can rewrite the constraint in (8) as follow.

$$\lambda_i n_i = \gamma_i - \bar{f}_i(p), \text{i=1,2,...,k} \tag{9}$$

The path length defined in (7) is called Normal measure length, or NM_length. Note that NM_length is nonlinear and the length of path p in this definition is not necessarily positive. (Due to page limitations, detailed discussions related to the intuitive meaning of λ_i and how different Pareto optimal points can be found by varying β had to be skipped here but can be found in [12].)

We now give some important properties of NM_length. Again due to page limitations, we omit their proofs but they can be found in [12].

Theorem 1. *If the length of a path p is larger than the length of routing constraint c (which is taken as a point in objective space), then at least one weight of the path p violates the corresponding constraint.*

Theorem 2. *Suppose two paths* $p(w_1, w_2, \ldots, w_k)$ *and* $q(w'_1, w'_2, \ldots, w'_k)$ *are given. If* $F(p) \preceq F(q)$*, then the length of path p is no longer than that of q.*

Theorem 3. *Using LPLF in a given search direction* $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$*, Dijkstra(*$\alpha$*) generates a Pareto optimal solution*

Theorem 4. *Let* $p(w_1, w_2, \ldots, w_k) \in P_{sd}$ *denote the solution generated by Dijkstra(*α*). If routing request* $c = c(c_1, c_2, \ldots, c_k)$ *dominates* $p(w_1, w_2, \ldots, w_k)$*, then c cannot be satisfied by any paths in* P_{sd}*.*

4.2 NM_MCP Algorithm

As outlined in Fig 1, the proposed algorithm NM_MCP mainly consists of two phases: precomputation and on-demand computation. In the precomputation phase, NM_MCP executes the procedure called Pre_computation given in Fig 2. This procedure simply computes the shortest paths individually with respect to each weight. We call those paths that minimize the individual weights as *primary paths*. Note that these paths depict the 'state' of the network from the viewpoint

```
NM_MCP($G, s, d, c$)
$G$: Network topology;    $s$: source node;
$d$: destination node;    $c$: routing request $(c_1, c_2, \ldots, c_k)$
$Boundlen$: The length of the routing request
    (1): $p^{i^*}$=Pre-computation($G$,$s$), $i = 1, 2, \ldots, k$
    (2): on-demand computation
        (2.1) Check_feasibility($p^{i^*}, c$)
        (2.2) if feasibility is not decided
            (2.2.1) Preparation($p^{i^*}, c = (c_1, c_2, \ldots, c_k)$)
            (2.2.2) NM_Dijkstra($G, s, d, c$,$Boundlen$)
```

Fig. 1. Proposed NM_MCP algorithm

```
(I)Pre_computation$(G, s)$
$G$: Network topology; $s$: Source node
(1): Computing $k$ primary paths $p^{i^*}$,for $i = 1, 2, \ldots, k$
(2): Determining Utopia point
        $F^* = [f_1^*, f_2^*, \ldots, f_k^*]^T = [f_1(p^{1^*}), f_2(p^{2^*}), \ldots, f_k(p^{k^*})]^T$
(3):Computing normalizing vector $L = [l_1, l_2, \ldots, l_k]^T$

(II)Check_feasibility$(p^{i^*}, c)$
(1)For $i = 1, 2, \ldots, k$
    (a) If $p^{i^*} \preceq c$      return $p^{i^*}$; success
    (b) If $c_i < w_i(p^{i^*})$return fail;
(2)End of for
(3)Otherwise, return undecided
```

Fig. 2. Procedure Pre_computation and Check_feasibility

of the source node s and will be used in on-demand computation phase. Based on the primary paths, we also obtain $f_i^*, i = 1, 2, \ldots, k$, and compute normalizing vector L using Eq. (6).

In the on-demand computation phase, which is entered when a routing request c arrives, NM_MCP first calls Check_feasibility that is also given in Fig 2.

This procedure checks the feasibility of primary paths. If there is no primary paths that can satisfy c, NM_MCP tries to find a feasible path by executing a modified version of Dijkstra's algorithm (called NM_Dijkstra) with respect to the new link weight NM_length. Recall that different values for β will lead to different shortest paths with respect to NM_lengh. So, one key issue here is, for a specific routing request c, how to select a proper β that can generate a better path. Actually, we need to determine $\bar{\phi}\beta$ rather than the specific values of $\bar{\phi}$ and β. For this, we take the intersection of the normal which is across the routing request $c = (c_1, c_2, \ldots, c_k)$ and the Utopia hyperplane.

To compute $\bar{\phi}\beta$ and other required parameters (e.g., the length *Boundlen* of the routing request which will be used as the upper bound for paths explored in NM_Dijkstra), NM_MCP calls Procedure Preparation given in Fig 3. It then executes NM_Dijkstra. Similar to the standard Dijkstra's algorithm, NM_Dijkstra maintains only three labels: cost(u), parent(u), and len(u) for each node u. cost(u) is a k-dimensional vector, each entry of which represents the individually accumulated link weights along the current path from s to u. parent(u) represents the predecessor of node u. len(u) is the length of the path from s to u. NM_Dijkstra initially sets len(u)=∞ and parent(u)=NIL for every node u. It then starts from node s, setting each entry of cost(s) to 0 and len(s) to $-\infty$. It then updates the labels of each adjacent node of s using the relaxation process shown in Fig 4.

```
Preparation($p^{i^*}, c = (c_1, c_2, \ldots, c_k)$)
(1): Calculating $\bar{\phi}\beta$
 (1.1)Determine the normal$\hat{n} = (n_1, n_2, \ldots, n_k)$ that across the routing request
 (1.2)Get the intersection point of $\hat{n}$ and the hyperplane $U$.
         This point is taken as $\bar{\phi}\beta$, and denoted by vector $(\gamma_1, \gamma_2, \ldots, \gamma_k)$
(2): Computing the length of routing request
 (2.1)Normalizing the routing request: $\bar{c}_i = \frac{c_i - f_i^*}{l_i}$
 (2.2)$\lambda_i = \frac{(\gamma_i - \bar{c}_i)}{n_i}$
 (2.3)Boundlen=-min($\lambda_i$)
```

Fig. 3. Preparation procedure

```
NM_Dijkstra_Relax(u,v)
(1) Increase=$(w_1(u,v), w_2(u,v), \ldots, w_k(u,v))$
(2) Tempcost=cost($u$)+Increase
(3) For $i = 1, 2, \ldots, k$
        newcost($i$)=$\frac{\text{Tempcost}(i) - f_i^*}{l_i}$
(4) For $i = 1, 2, \ldots, k$
        $\lambda_i = \frac{(\gamma_i - \text{newcost}(i))}{n_i}$
(5) templen=-min($\lambda_i$)
(6) If templen < len($v$)
        len($v$)=templen;  parent($v$)=$u$;  cost($v$)=Tempcost
(7) EndIf
```

Fig. 4. Relaxation procedure of NM_Dijkstra

This relaxation procedure is the fist modification to the Dijkstra's algorithm. NM_Dijkstra then continues to explore the graph by choosing the next node that has the least length. The second modification here is to consider only those nodes whose lengths are smaller than the Boundlen when deciding whether to extend a node or not.

The computational complexity of NM_MCP algorithm is (k+1) times that of the Dijkstra's algorithm, k times for precomputation and (if needed) one time for on-demand computation. Therefore, the worst-case response time of NM_MCP is at most one iteration of the Dijkstra's algorithm while the best of existing on-demand algorithms require at least a few iterations of it.

4.3 Improvements to NM_MCP

Since NM_MCP adopts nonlinear path length, it also suffers from the inherent drawback of nonlinear path lengths, namely the sub-section of a shortest path is not necessary the shortest one. To further improve the performance, we enhanced NM_MCP by using a look-ahead method similar to the one used in [15]. In the look-ahead method, we first compute the shortest path tree rooted at the destination to each node v in the graph. For this, Dijkstra's shortest path algorithm is executed k times for each link weight separately. Accordingly, the least length between the destination node and every node u is determined and maintained with respect to each link weight individually. We then use this information in the relaxation process of NM_Dijkstra algorithm to have a better estimation of the path length based on NM_length. The computation complexity of NM_MCP with look-ahead ability is $(2k+1)$ times that of Dijkstra's algorithm. To distinguish from Pareto look-ahead, we call this look-ahead as nonlinear look-ahead.

5 Performance Evaluation

Performance evaluation includes three parts. The first part is the comparison of NM_MCP with H_MCOP [16][11], which is an efficient on-demand QoSR algorithm using nonlinear path length, and MEFPA [8], which is an efficient precomputation algorithm using linear path length. The second part evaluates the performance increase introduced by Pareto look-ahead. The last part evaluates the response time of NM_MCP.

5.1 Simulation Model and Performance Measures

Topologies used for simulations are based on Waxman's model with 50, 100 and 200 nodes. Each link is associated with k weights using $w_i \sim$ uniform[1,300]. For each node number, 200 graphs are generated per experiment. For each instance of a random graph, one source node and 25 destination nodes, which are at least two hops away from the source node, are generated randomly. For each source-destination pair, one routing request is generated.

As the key performance measure, we use success rate (SR), the ratio of the routing requests satisfied by heuristic to the total routing requests generated.

5.2 Performance Comparison of QoSR Algorithms

In this part, we compare the SR of NM_MCP with that of two QoSR algorithms: H_MCOP and MEFPA. The computational complexity of H_MCOP for every routing request is twice that of the Dijkstra's algorithm. MEFPA, on the other hand, requires pre-iterations of the Dijkstra's algorithm, where b is a user specified parameter that controls the complexity and the performance. MEFPA with parameter b is denoted by MEFPA(b). We use NM_MCP2 to denote NM_MCP with nonlinear look-ahead ability.

Routing requests are generated with respect to the k primary paths denoted by $p_i(w_{i1}, w_{i2}, \ldots, w_{ik}), i = 1, 2, \ldots, k$. Let $f_j^{max} = \max(w_{ij})$ and $f_j^{min} = \min(w_{ij}), i = 1, 2, \ldots, k$.

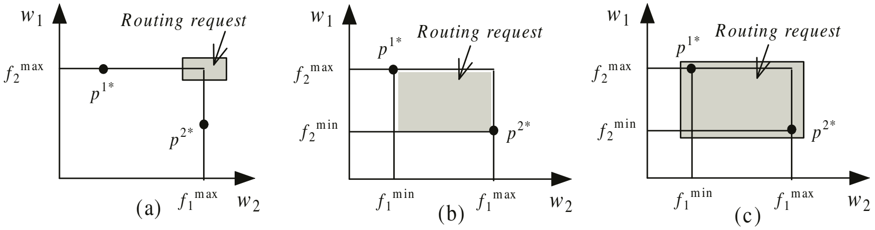


Fig. 5. The distribution of critical routing requests

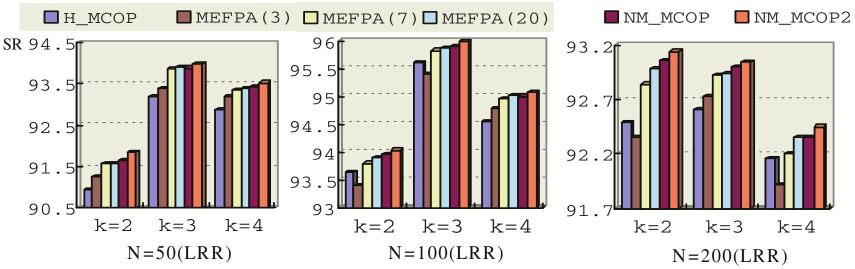


Fig. 6. Performance comparison of different algorithms under loose routing requests

We first consider the same method in [16] to generate routing requests. Specifically, we generate:

$$c_i \sim \text{uniform}[0.8 * f_i^{max},\ 1.2 * f_i^{max}] \tag{10}$$

The case of generating constraints according to (10) in bi-objective QoSMS is shown in Fig 5.(a). As seen from the figure, most routing requests generated in this manner are likely to be feasible. We call such routing requests as *loose routing requests* (LRR).

In fact, the most critical routing requests lie in the area shown in Fig 5.(b) [9]. This critical area is the so called “NP-complete” range [17]. Loose routing requests only cover a small part of this area. So we use another method to generate routing requests:

$$c_i \sim \text{uniform}[0.8 * f_i^{min},\ 1.2 * f_i^{max}] \tag{11}$$

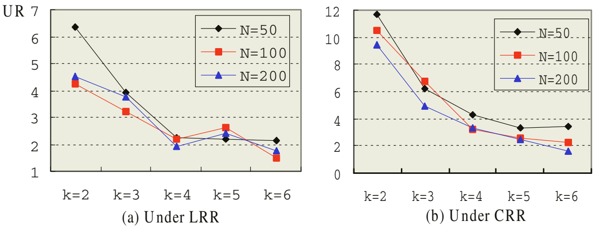
Fig 5.(c) shows the case of generating constraints according to (11) in bi-objective QoSMS. The whole NP-complete range is covered by the distribution of routing requests. We call such routing requests as critical routing requests (CRR).

Fig 6 show the performance comparison of various algorithms under multiple constraints using LRR. As clearly seen from the figures, NM_MCP gives high SR and MEFPA becomes comparable to it when b is increased. When b takes a larger number, MEFPA(b) will introduce not only a high computation overhead but also a larger routing table (after each iteration of Dijkstra's algorithm, a path will be stored in routing table) [9]. On the other hand, NM_MCP only performs k iterations of Dijkstra's algorithm and give comparable SR to that of MEFPA(20). This implies that NM_MCP can re-execute the Pre_computation procedure more frequently than MEFPA within the same period. As a result, NM_MCP will suffer less from stale routing information and give better performance in practice.

Due to their ability to cope with stale routing information, on-demand computation is also essential for the success of QoSR solutions. But H_MCOP requires two iterations of Dijkstra's algorithms. However, NM_MCP executes Dijkstra's algorithm at most once and give better performace with the help of the precomputed paths. Hence, NM_MCP has a quicker response speed than H_MCOP. So NM_MCP makes a good tradeoff between precomputation and on-demand computation. Furthermore, NM_MCP uses LPLF in precomputation phase and NLPLF in on-demand computation phase, making a good tradeoff between LPLF and NLPLF as well.

5.3 Evaluation of Pareto Look-Ahead

In this part, we evaluate the performance increase brought by Pareto look-ahead mechanism. We use the unfeasible rate (UR), the ratio of the times that on-demand computation is avoided to the number of total routing requests generated. Simulation results for UR versus k (which denote the number of link wights) are shown in Fig 7. We can see that performance gains are less as k increases. This is because when k is large, the probability that routing requests dominate primary paths is small. Moreover, when routing requests are critical, considerable parts of them are inherently unfeasible. So the probability that these routing requests dominate primary paths is large and hence more performance increase is gained.



(a) Under LRR

(b) Under CRR

Fig. 7. Evaluation of Performance increase brought by Pareto look-ahead

5.4 Evaluation of Response Time

A practical QoSR algorithm should have not only a high SR, but also a quick speed of response. In this part, we look at how often routing requests can be

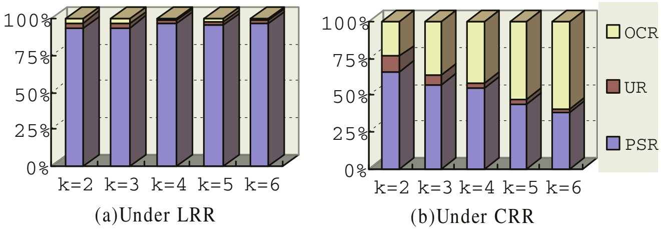


Fig. 8. Evaluation of response time and on-demand computation overhead

responded immediately by NM_MCP. Routing requests that can be responded immediately includes those satisfied by primary paths and those determined to be unfeasible by Pareto look-ahead. Hence we count individually the precomputation success rate (PSR), the ratio of the number of routing requests satisfied by primary paths to the number of routing requests generated, and UR. Routing requests that cannot be responded immediately need further on-demand computation. We count the ratio of the number of these routing requests to the number of total routing requests as on-demand computation rate (OCR).

Fig 8 shows the simulation results when routing request are loose and critical with node number N=200. We can see that when routing requests are loose, most routing requests can be responded immediately and only very small parts of routing requests need further on-demand computation. When routing requests are critical, again a considerable part of routing requests does not require further on-demand computation. Note that only one iteration of Dijkstra's algorithm is executed for each routing request that needs on-demand computation. On average, on-demand computation overhead of NM_MCP is significantly lower than one iteration of Dijkstra's algorithm, particularly under LRR.

6 Conclusions

To solve a given MCP problem, we derived from it a new MCMOP problem, which can be viewed as a special case of discrete version of MOOP. We considered this problem to develop a new nonlinear path length function, namely NM_length. Based on this length function, we designed an efficient algorithm (namely, NM_MCP). Using extensive simulations, we showed that NM_MCP algorithm is very efficient when both SR and response time are taken into account.

References

1. Korkmaz, T., Krunz, M.: Bandwidth-delay constrained path selection under inaccurate state information. IEEE/ACM Transactions on Networking (ToN) **11** (2003) 384–398
2. Korkmaz, T.: State-path decoupled QoS-based routing framework. In: Proceedings of the IEEE GLOBECOM '04 Conference. (2004) (to appear).

3. Chen, S., Nahrstedt, K.: An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. IEEE Network **12** (1998) 64–79
4. Zheng, Y., Dou, W., Tian, J., Xiao, M.: An overview of research on qos routing. In: Advanced Parallel Processing Technologies(APPT03), Springer LNCS(2834) (2003) 387–397
5. Kuipers, F., Korkmaz, T., Krunz, M., Van Mieghem, P.: Performance evaluation of constraint-based path selection algorithms. IEEE Network (2004) (to appear).
6. Jaffe, J.M.: Algorithms for finding paths with multiple constraints. Networks **14** (1984) 95–116
7. Juttner, A., Szviatovszki, B., Mecs, I., Rajko, Z.: Lagrange relaxation based method for the QoS routing problem. In: Proceedings of the INFOCOM 2001 Conference. Volume 2., IEEE (2001) 859–868
8. Cui, Y., Xu, K., Wu, J.: Precomputation for multi-constrained qos routing in high-speed networks. In: Proceedings of the INFOCOM 2003 Conference, San Francisco (2003)
9. Zheng, Y., Tian, J., Liu, Z., Dou, W.: An efficient dynamic weight coefficient qos routing algorithm. In: International Network Conference(INC04), UK (2004)
10. De Neve, H., Van Mieghem, P.: TAMCRA: a tunable accuracy multiple constraints routing algorithm. Computer Communications **23** (2000) 667–679
11. Korkmaz, T., Krunz, M.: Routing multimedia traffic with QoS guarantees. IEEE Transactions on Multimedia **5** (2003) 429–443
12. Zheng, Y., Korkmaz, T., Zhang, H., Dou, W.: Pre- and on-demand computations along with a new normal measure. Technical report (2004) http://www.cs.utsa.edu/∼korkmaz/yanxing/TR-NM-MCP.pdf.
13. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim **8** (1998) 631–657
14. Messac, A., Ismail-Yahaya, A., Mattson, C.: The normalized normal constraint method for generating the pareto frontier. Struct. Multidisc. Optim **25** (2003) 86–98
15. Korkmaz, T., Krunz, M.: A randomized algorithm for finding a path subject to multiple QoS constraints. Computer Networks Journal **36** (2001) 251–268
16. Korkmaz, T., Krunz, M.: Multi-constrained optimal path selection. In: Proceedings of the INFOCOM 2001 Conference. Volume 2., Anchorage, Alaska, IEEE (2001) 834–843
17. Kuipers, F., Mieghem, P.: The impact of correlated link weights on qos routing. In: Proceedings of the INFOCOM 2003 Conference, IEEE (2003)

An Architecture for Software-Based iSCSI: Experiences and Analyses

Annie Foong, Gary McAlpine, Dave Minturn,
Greg Regnier, and Vikram Saletore

Intel Corporation, 2111 NE 25th Ave, Hillsboro, OR 97124
{annie.foong, gary.l.mcalpine, dave.b.minturn,
greg.j.regnier, vikram.a.saletore}@intel.com

Abstract. Supporting multi-gigabit/s of iSCSI over TCP can quickly saturate the processing abilities of a SMP server today. Legacy OS designs and APIs are not designed for the multi-gigabit IO speeds. Most of industry's efforts had been focused on offloading the extra processing and memory load to the network adapter (NIC). As an alternative, this paper shows a software implementation of iSCSI on generic OSes and processors. We discuss an asymmetric multiprocessing (AMP) architecture, where one of the processors is dedicated to serve as a TCP engine. The original purpose of our prototype was to leverage the flexibility and tools available in generic systems for extensive analyses of iSCSI. As work proceeded, we quickly realized the viability of generic processors to meet iSCSI requirements. Looking ahead to chip-multiprocessing, where multiple cores reside on each processor, understanding partitioning of work and scaling to cores will be important in future server platforms.

Keywords: iSCSI, Asymmetric Multiprocessing, TCP optimization.

1 Motivation

The concept of Internet Protocol (IP)-everywhere is appealing – the same infrastructure and expertise can be deployed across all networks. Additionally, the economies of scale that is available through commodity IP networking infrastructure (GbE adapters, switches, cables) makes deployment affordable. While IP network file-based protocols (e.g. NFS, CIFS) had been around for years, many enterprise workloads, such as database systems and high definition streaming video, are optimized for direct access to block storage. The recently defined iSCSI (SCSI over IP) standard is an alternative to currently deployed FC-based storage area networks (SAN) and offers the potential of an IP-converged SAN. Although iSCSI is transport-agnostic, a current workable implementation is most likely to be deployed over TCP. To achieve IP-converged cluster deployments, the performance and scalability of iSCSI must approach that of FC SANs. We recognized (and shall show in this paper), that the major overhead in iSCSI is not iSCSI itself, but TCP. In the case of FC, the bulk of protocol processing is offloaded to hardware on the host-bus adapters (HBAs).

R. Boutaba et al.(Eds.): NETWORKING 2005, LNCS 3462, pp. 65 – 77, 2005.

As such, implementers in the IP space have envisioned that a reasonable solution would be a similar hardware-assisted approach, and offload TCP and/or iSCSI to TCP offload engines (TOEs) and iSCSI HBAs respectively. However, hardware implementation is difficult and fraught with errors [17]. Interfaces between host and engines are crucial, but are typically not well understood [12]. Finally, TCP has a far more complex state machine than other transports. Unlike FC which is designed specifically for hardware implementation from ground up, TCP began life as a software stack. Corner cases abound that are not so easily addressed if the solutions are hardwired. Researchers [12, 17] also reasoned that the complex NIC chips often lag behind the performance of generic processors that tend to ride Moore's law.

We, therefore, chose to focus on an architecture for iSCSI in *software*. Three major trends motivated our direction: (1) Commercially viable chip multiprocessors (CMP), where increased processing power is achieved through multiple cores per CPU, rather than clock speed ramp [8]; (2) Integration of the memory controller on the CPU die will effectively scale memory bandwidth with processing power; (3) Huge strides in bus bandwidth improvement. (2) & (3) can potentially remove the chipset and bus as system bottlenecks. The availability of many cores in CMP can potentially breach the proverbial cpu-memory gap if software is optimally re-architected for thread-level parallelism. To effectively leverage many cores for networking becomes the key to designing future CMP servers.

2 Approach

Network protocol stacks of general-purpose monolithic operating systems, are known for their inability to scale well in SMPs [11, 22]. We take an asymmetric multiprocessing (AMP) approach, where one of the processors is dedicated to serve as a TCP engine. By separating the protocol processing from the rest of the operating system (OS), we hope to provide a clearly-defined sandbox whereby protocol processing can progress independent of limitations incurred by generic OSes. Additionally, we incorporated well-known network optimization techniques, including zero copy and asynchronous interfaces. We recognize the impact of effective application interfaces on network performance. In particular, existing iSCSI implementations [21] use BSD-style sockets, and force SCSI (which is inherently asynchronous) to be built upon synchronous interfaces. We therefore took a full iSCSI stack implementation and re-architected it from ground up. We modified a reference implementation of iSCSI to use an asynchronous sockets-like API that we built. This allowed us to perform an experimental-based evaluation of our APIs and overall architecture as compared to current practices. Research questions of interest to us in this study are: *1) What is the processing requirements of iSCSI in SMP mode ? 2) What parts of iSCSI processing can benefit from the AMP model ? 3) What constitutes an optimal architecture for software-based iSCSI ?*

We give a brief overview of our software-based architecture in Section 3. In Section 4, we discuss our experimental work and how iSCSI uses asynchronous interfaces. In section 5, we present in-depth performance analyses of iSCSI processing requirements on SMP Linux and our AMP prototype. By separating the stack into functional bins, we are able to pinpoint exactly where and why the AMP model makes a difference. Wherever possible, we share the hands-on experiences and lessons learnt from this work. We conclude with related and future research.

3 Overview of Architecture

Full architectural details of our AMP prototype were described in [16]. In that work, our prototype was functional to a point where limited runs of one-way bulk data transfers was possible. The work presented here is work done well beyond that phase. Fig 1(a) shows a typical iSCSI implementation over generic (2P) SMP Linux. The image footprint is duplicated symmetrically across the 2 processors. Our iSCSI/AMP architecture takes a hard affinity, asymmetric viewpoint whereby application and network processing are partitioned along very deliberate lines (Fig 1b). We refer to the 'host' as the processor where the generic OS and applications reside. The 'packet processing engine' (PPE) is where network protocol processing (i.e. TCP/IP) is performed. The PPE in our prototype is a loadable Linux module consisting of only the TCP/IP stack. Once inserted, the PPE goes into a poll loop and never yields the processor. Here lies the crux of our AMP design: The hard partitioning allows the PPE to continuously poll for work from the NICs or the host. *The PPE does not get interrupted by devices, nor scheduled by the host OS.* The PPE runs without interference from the OS and devices. The poll is performed on shared memory. The PPE can poll NIC descriptors for synchronization, without causing memory bus traffic, until the cache-lines of the associated shared memory is modified. Finally, since the PPE determines its own path of execution, it can predict memory usage and pre-fetch accurately. The host to PPE interface is implemented as a set of asynchronous queues [4] in cache-coherent, shared host memory. The doorbell queue is emulated in software and is the mechanism for the host to inform the PPE that work has been posted on the work queues.

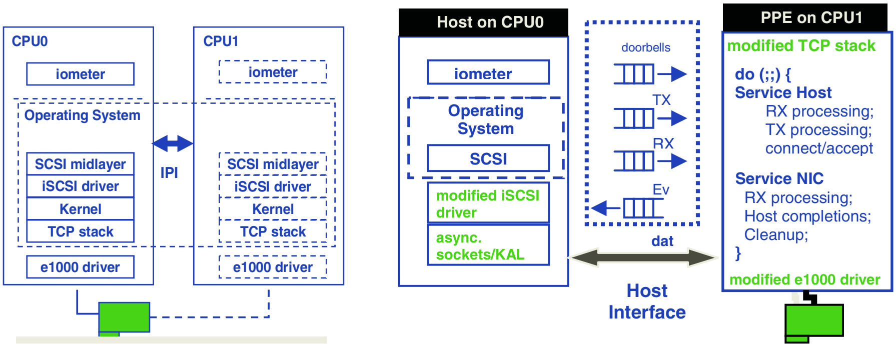


Fig. 1. (a) iSCSI in SMP mode (Linux) (b) iSCSI in AMP mode

Conversely, the event queue is a means for the PPE to inform the host that work has been completed. Furthermore, we have specifically designed our host-engine interface to remove locking. Our queues are one-way (e.g Host only posts to work queues; PPE reads from work queues). This explicit producer-consumer model completely eliminates locking between PPEs and hosts. In the general case with multiple hosts and PPEs, locking will only be required between host to host, and PPE to PPE. Looking ahead to Receive-side scaling enabled NIC implementations [22], which have the ability to parse and direct flows to specific processors/cores, there will be minimal need for locking among PPEs. The programming interface is an asynchronous, sockets-like interface, over which we have built over iSCSI implementation. A Kernel Adaptation Layer (KAL) provides the necessary shim that hides the innards of work and event queues manipulation from the host application. This interface also enables pre-posting, and is essential to the implementation of zero-copy. There is true zero-copy on the transmit path (loosely based on *tcp_sendpage()* used by in-kernel applications in Linux); and zero-copy from the host's point of view on the receive path. The PPE does the copy on behalf of host.

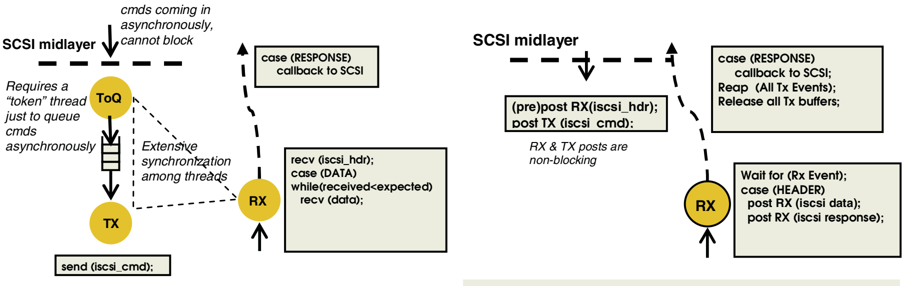
4 Implementation Details

4.1 Lessons in Experimental Setup

Our initiator is a system with 2P Intel® Xeon™ 2.4GHz processors on a 400 MHz front-side bus. The OS is the Linux-2.4.18 kernel. 2 standard Intel Pro1000 GbE controllers, on a 64bit/133MHz PCI bus, are used to match the performance of 2Gbps FC. The base iSCSI initiator/target code is from SourceForge [21]. We used IOmeter [19] to exercise read/write transactions on the SCSI layer. We modified dynamo (IOmeter's driver) to use raw IO in order to turn off buffer caches on the initiator. Operating in this mode forces a serialization of requests in the SCSI midlayer. Because of various serializations throughout a complex system typical of any iSCSI setup, we went through great lengths to ensure that bottlenecks, other than iSCSI processing, are removed. We ran 2 instances of dynamo to parallelize block requests and get around Linux's SCSI serialization. We configured IOmeter and our iSCSI initiator driver to issue multiple outstanding IO commands. This functionality is necessary to circumvent the roundtrip latencies of TCP. However, the iSCSI targets we used did not have such support. We emulated such support through over-provisioning socket buffers (set to 128KB) on our targets. This workaround worked for our particular one-initiator (one connection) to on-target setup. Finally, we used RAM disks on the targets to remove dependencies on disk IO speeds.

4.2 iSCSI's Usage of Asynchronous APIs

iSCSI encapsulates SCSI command data blocks into TCP/IP packets that can be sent over IP networks [21]. The iSCSI protocol is an end-to-end transaction-based protocol, very much like SCSI. It goes through phases of authentication and discovery before entering the full feature phase where data is transferred. iSCSI is typically implemented as a low-level driver, and may be called in interrupt context. Polling for events would be inefficient, and explicit blocking is not an option. Since processors must not block waiting for disks, the SCSI midlayer is asynchronous. However, most iSCSI implementations are built over BSD synchronous sockets, and require additional threads to emulate a non-blocking iSCSI driver **Fig. 2**(a). At least 2 threads are required per connection. Substantial synchronization among threads will become an issue as the number of connections increases. Asynchronous APIs (**Fig. 2**(b)) simplifies the picture. Transmits are posted asynchronously without the need for a token thread. We have used a receive thread (RX) in this prototype version. The final version, using an event callback mechanism, removes the need for a RX thread altogether. An asynchronous interface allows our iSCSI implementation two ways of waiting for an event: (1) by *polling* the event queue or (2) by *explicit blocking* (sleeping) if there are no pending events. In the latter case, an inter-processor interrupt (IPI) is generated only if it is necessary to wake the sleeping process. The ability to either poll or sleep is a powerful tool that allows the user to control its own response to events.


SCSI midlayer
cmds coming in asynchronously, cannot block
Requires a "token" thread just to queue cmds asynchronously
ToQ
Extensive synchronization among threads
TX
RX
SCSI midlayer
RX & TX posts are non-blocking
RX

```
case (RESPONSE)
   callback to SCSI
```

```
recv (iscsi_hdr);
case (DATA)
while(received<expected)
   recv (data);
```

```
send (iscsi_cmd);
```

```
(pre)post RX(iscsi_hdr);
post TX (iscsi_cmd);
```

```
case (RESPONSE)
   callback to SCSI;
Reap  (All Tx Events);
Release all Tx buffers;
```

```
Wait for (Rx Event);
case (HEADER)
 post RX (iscsi data);
 post RX (iscsi response);
```

Fig. 2 (a). iSCSI reads over synchronous APIs (b) asynchronous APIs

While iSCSI HBA implementation deals with entire requests, our software-based iSCSI has an inherent byte-stream nature to its user interface (both synchronous and asynchronous versions of sockets). The non-existent message boundary makes it difficult to determine the granularity of events (i.e. when work is considered as done). Along similar lines, we noted in our work the awkwardness of pre-posting for reads without a direct data placement [23] awareness in the PPE. During an iSCSI read, we can at best pre-post for the reading of the iSCSI header (typically 48B). It is on parsing the header contents that we know how much more to receive. If direct data placement is available, the initiator would have pre-posted enough buffers to handle the headers, data payloads, and command statuses. On correct placement of all packets, the PPE would issue 1 event that signals the completion of an entire request.

In our particular implementation, we emulated this behavior by limiting our initiator to communicate with only one target per connection. We depended on the socket buffers as the command and data pipeline, so that throughput does not suffer. In this way, we can pre-post buffers, for both the header and data, at command issue time. In our case, we know that whatever comes back is for the current outstanding command. This is not generally applicable since responses (i.e. associated data) to commands can, and will come out of order. Even with this optimization, there is not an appreciable improvement. The *wait()* semantics that our prototype implemented only allowed for the reaping of one event. Every posting of work must be accompanied by a corresponding *wait()* (or *poll()*). In the worst case, this may translate to an interrupt from the PPE for every event. Working with iSCSI exposed the need for a *wait_N()* functionality (i.e. wait for N events before waking the host). With *wait_N(),* an application can better coalesce events according to what was previously posted. E.g. *Wait_2()* would wake the host only on the arrival of both headers and data. The earlier issue of byte-stream semantics still haunts us in the PPE, and we are unable address it unless a framing protocol [23] is added to TCP.

5 Performance Analysis

In this section, we present a series of results comparing the performance of iSCSI over generic SMP Linux and our AMP prototype for sequential iSCSI reads/writes of 8KB and 64KB requests. We also noted frequency scaling characteristics to project to future processors. Finally, we used Oprofile (a profiler based on event sampling) [20] to determine the breakdown of processing hotspots and perform in-depth analyses to fully understand performance impact and provide reasons behind them.

5.1 Throughput and Utilization

Fig 3(a) shows improved bandwidth and reduced CPU utilization of AMP over SMP. Worthy of note is the AMP prototype's ability to push throughput at line rate (~200MB/s). A more illuminating view is to look at the cost metric of GHz/Gbps (i.e. cycles per bit transferred) in Fig 3(b). The AMP model is 30-40% more efficient than SMP. The ability (or inability) to scale across processor frequencies exposes dependencies on non-processing components (Fig 4).

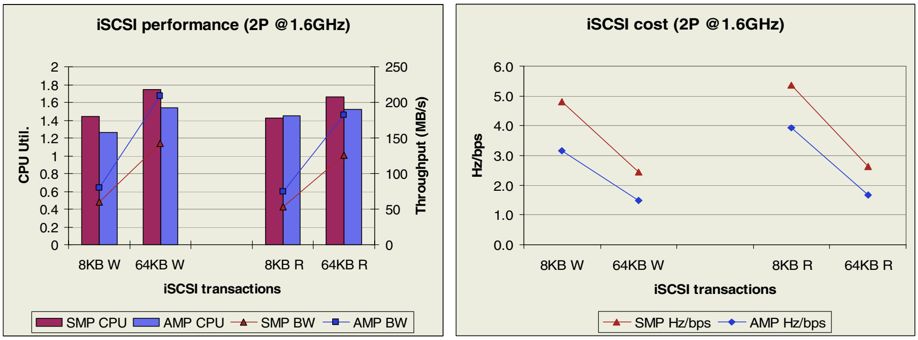


Fig. 3. (a) iSCSI throughput & CPU utilization (2P full utilization is 200%) (b) iSCSI cost

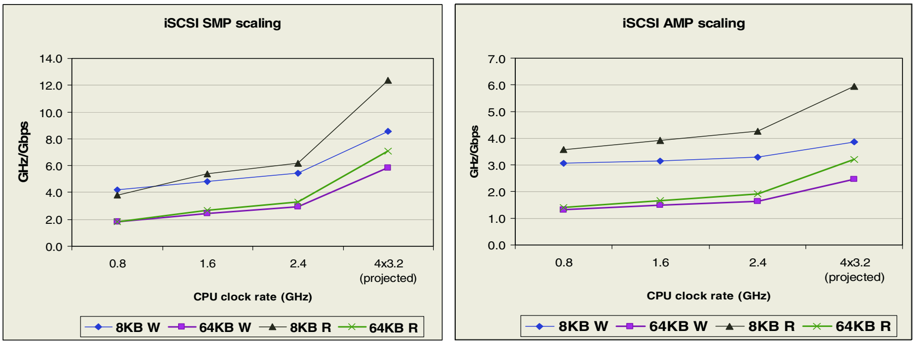


Fig. 4. SMP versus AMP scaling (*the flatter than lines, the better the scaling*)

5.2 Hotspot Profiles

The SMP model implies that the image footprint is duplicated exactly across processors. However, it does not imply exact load balancing. In Windows NT and Linux default configurations, interrupts go only to CPU0 [2, 22]. Scheduling is dependent not only on running processes, but also on interrupts coming in from devices. (**Fig 5, Fig 6**) give a visual representation of iSCSI processing distribution for the 2 models. Exact percentages will be given in the next section. We have carefully binned Linux functions into logical layers and abstracted processing to a level where analysis gave useful insights. As much as we can, we have separated the compute-intensive parts of TCP protocol processing (*TCP*), i.e. the cranking of the state machine, from the kernel support, memory-intensive parts of TCP processing (*kernel*). Kernel support includes memory/buffer management routines, the manipulation of TCP contexts, timers, etc. A full implementation of the sockets interface includes not only the obvious BSD sockets API, but also system calls, and schedule-related routines. This is how an application causes a socket action to be executed from the user level all the way down to the TCP stack. We put all these functions into the *schedule* bin. *Interrupts* refer to NIC interrupt processing, which logically also belongs to interfaces. We have separated them here to showcase that our PPE does not take device interrupts. For AMP, *q-processing* roughly takes the place of interrupts and system calls. *Driver* refers to the NIC driver routines. *Others* refer to miscellaneous user routines including runtime libraries and Oprofile. Finally, *copies* are of movement of payload data. This allows us to contrast the one-copy path of standard OS with our prototype's zero-copy implementation on writes. We have established that the major hotspots of iSCSI processing are in *SCSI*, *TCP* engine, *kernel* support and *copies*. iSCSI (without data integrity computation) takes less than 5% of processing. Because of Linux 2.4 use of a single io_request_lock spinlock for the entire block device system, the SCSI midlayer makes up a much larger overhead.

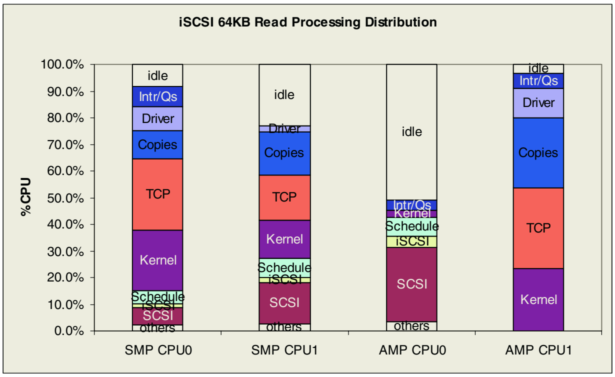


Fig. 5. Profile comparison of iSCSI 64KB reads

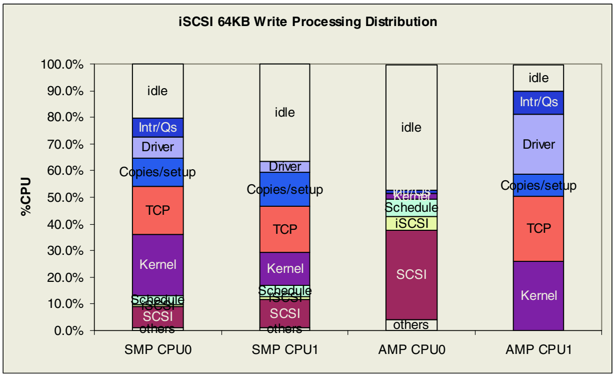


Fig. 6. Profile comparison of iSCSI 64KB writes

5.3 In-depth Analyses

Intuitively, the most notable benefit of hard partitioning is improved cache locality. In this section, we will quantify that statement, and pinpoint exactly where and why improvements occur as we go from the SMP to AMP mode. Also, we have combined interrupts, system calls, queue processing and scheduling appropriately into the *interface* bin. In addition to counting cycles (time), we also noted cache misses and machine clears, as they tend to have the largest performance impact [7].

We begin by first getting a per-CPU view of iSCSI processing (Table 1). All counters have been normalized to work done. There is a reduction of cycles per instruction (CPI) in all bins, between SMP and AMP modes, showcasing the overall higher efficiency in AMP processing. As expected, the *others* and *SCSI* layers, which are untouched, show similar counts in both modes. We believe it is useful to call out that the %distributions by themselves cannot reveal this insight, an evaluation of absolute counts is necessary. Due to space limitation, we will call out analyses using writes as an example (reads can be similarly interpreted)). *TCP* protocol processing has a CPI of 3.7-4.5 for SMP, and 3 on the PPE for AMP. *Kernel* takes 22.5% on CPU0 and 12.3% on CPU1 for SMP. The different loading on CPU is due to the glaringly large difference in the number of machine clears. NIC interrupts go to CPU0, in addition to IPIs from CPU1. *Kernel* support overheads are primarily on the PPE for AMP, once again with much improved CPI. iSCSI "writes" incurs no copies, but do require setup for zero-copy. Zero-copy had essentially transformed *copies* from a "memory-intensive" bin (note cache misses in SMP W) to a "compute-intensive" bin (note number of instructions in AMP W). We should point out that copies on read (under Linux-2.4.x) are implemented via `rep movl` (repeat string moves) which explains the large CPIs (18.7-21.9). Copy for writes is a carefully crafted rolled-out loop that moves data efficiently based on alignment that is known beforehand. Reads were implemented assuming arbitrary arrival of bytes. An optimized version of copy on read had since appeared in Linux-2.6 [18]. The largest improvement in CPI is seen in *interface* (SMP: 10.5-15.1, AMP:1.3-5.2). Once again, the large number of machine clears on CPU0 is notable. Interestingly, *interfaces* do not contribute to large processing hotspots in themselves. But, they have an indirect impact on pipeline and caching behaviors. These overheads are the price a software network stack pays for existing in a generic OS environment. They represent *intrusions* points into the processing path other than themselves, and are tricky to characterize. We found machine clears to be a reasonable counter to quantify such impact. Machine clears (i.e. instruction pipeline flushes) are caused by context switches due to interrupts (e.g. from devices, IPIs, page faults, etc) and scheduling. Characterization of machine clears, therefore, allows us to get a handle on these elusive intrusion points. Evidence of their impact surfaces wherever large number of machine clears occur.

Finally, to provide a *quantifiable value* to our discussions, we combined total processing requirements on both processors and perform a speedup analysis (Table 2). For example, to derive % improvement in the number of machine clears (or other counters) in the *TCP* engine, when going from the SMP to AMP mode is obtained as follows:

$$\%\ Improvement = (clears\text{-}TCP_{SMP} / clears\text{-}total_{SMP}) \times (1 - clears\text{-}TCP_{AMP} / clears\text{-}TCP_{SMP})$$

$clears\text{-}TCP_{SMP}$ = Number of machine clears (per work done) observed in *TCP* on SMP
$clears\text{-}TCP_{AMP}$ = Number of machine clears (per work done) observed in *TCP* on AMP
$clears\text{-}total_{SMP}$ = Total number of machine clears (per work done) observed on SMP

Wherever we see an improvement in cycles, we also see corresponding improvements in cache misses and machine clears on AMP. There is a 11% (*out of a total of 34.6%*) improvement in cycles in *TCP*. Zero copy accounts for 9.4%, showcasing the importance of this optimization. Another 10.2% comes from improvement in *kernel* support. The routines used in *kernel* and *TCP* are essentially the same in either SMP or AMP modes. The improvements come from fewer cache misses in AMP mode. Despite our best efforts, *TCP* still contain memory-related routines (e.g. reading of TCP contexts when calculating window size). The elimination of interrupts and scheduling on the PPE resulted in much fewer machine clears (14.4% fewer in *kernel*, 15.9% fewer in *TCP*, out of a total of 46.8%). Since host and PPE processing are confined to the same processor on the AMP, the number of IPIs needed for synchronization is also reduced. The reason for the improvement seen in *copy* for reads is worth pointing out. In SMP, the bottom and top halves of the TCP stack can potentially be executed on any processor. Copies can incur cache misses on both source and destinations. In the AMP case, sources are always on the PPE, and destinations on the host. Looking back at the per-CPU breakdown in Table 1, we confirmed that the reduction in cache misses is primarily on the host in AMP. Functional partitioning had enabled destinations of *copy* to remain warm in cache.

Table 1. Per CPU characterization of SMP and AMP (normalized)

	CPU0						**CPU1**					
SMP W	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**
idle	8104615	506010	6424	114	20.4%		14580192	1024038	10114	481	36.8%	
others	359423	185096	0	1202	0.9%	1.9	545192	216346	319	956	1.4%	2.5
SCSI	3130769	1086538	3462	2332	7.9%	2.9	3987500	1614183	4345	2374	10.1%	2.5
iSCSI	310385	64904	126	1022	0.8%	4.8	356731	108173	204	913	0.9%	3.3
Interface	4338654	287260	30349	4381	10.9%	15.1	1696154	161058	4537	3840	4.3%	10.5
Kernel	8938462	1676683	24351	11563	22.5%	5.3	4874615	1080529	8395	9531	12.3%	4.5
TCP	7068077	1914663	22260	8347	17.8%	3.7	6935000	1540865	43882	12332	17.5%	4.5
Copies	4142885	1304087	6911	16352	10.4%	3.2	5012115	1657452	7530	19159	12.6%	3.0
Driver	3299423	638221	6683	9099	8.3%	5.2	1660769	371394	4724	2813	4.2%	4.5
Total Non-idle	31588077	7157452	94141	54297			25068077	6750000	73936	51917		
	CPU0						**CPU1**					
AMP W	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**
idle	9109619	681548	9458	122	47.0%		2120190	732738	2036	33	10.1%	
others	786476	290476	738	268	4.1%	2.7	0	0	0	0	0.0%	
SCSI	6494857	2534524	7854	991	33.5%	2.6	0	0	0	0	0.0%	
iSCSI	988857	339286	878	446	5.1%	2.9	0	0	0	0	0.0%	
Interface	1587333	305357	3911	1402	8.2%	5.2	1878762	1429762	1967	991	8.9%	1.3
Kernel	217714	45238	167	229	1.1%	4.8	5537905	1716667	5560	2220	26.3%	3.2
TCP	198667	63095	280	33	1.0%	3.1	5065619	1670238	6152	4875	24.0%	3.0
Copies	0	0	0	0	0.0%		1712667	757143	2616	286	8.1%	2.3
Driver	0	0	0	0	0.0%		4773810	1747619	4768	4970	22.6%	2.7
Total Non-idle	10273905	3577976	13827	3435			18968762	7321429	21063	13342		
	CPU0						**CPU1**					
SMP R	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**
idle	2560159	197421	2103	0	7.9%		7073492	543651	4851	233	21.9%	
others	708254	138889	714	2078	2.2%	5.1	824762	255952	908	1736	2.6%	3.2
SCSI	2165079	657738	2307	1781	6.6%	3.3	5011429	1983135	6156	3775	15.5%	2.5
iSCSI	519524	158730	352	957	1.6%	3.3	647302	230159	521	1161	2.0%	2.8
Interface	4152222	262897	32480	4067	12.7%	15.8	2271905	263889	6161	5427	7.0%	8.6
Kernel	7323333	1277778	18948	10665	22.5%	5.7	4677302	671627	13661	8348	14.5%	7.0
TCP	8747937	1897817	30303	9742	26.9%	4.6	5586667	1093254	34157	10342	17.3%	5.1
Copies	3507619	187500	4866	19286	10.8%	18.7	5331905	243056	7004	28482	16.5%	21.9
Driver	2886032	575397	9792	8373	8.9%	5.0	821270	173611	3249	1920	2.5%	4.7
Total Non-idle	30010000	5156746	99762	56949			25172540	4914683	71815	61190		
	CPU0						**CPU1**					
AMP R	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**	**cycles**	**instr**	**clears**	**L2 misses**	**%CPU**	**CPI**
idle	9872667	701190	10208	170	51.1%		737333	268452	673	0	3.5%	
others	660762	289881	551	229	3.4%	2.3	0	0	0	0	0.0%	
SCSI	5420762	2247024	6446	976	28.1%	2.4	0	0	0	0	0.0%	
iSCSI	779048	260119	771	473	4.0%	3.0	0	0	0	0	0.0%	
Interface	2100667	346429	7649	2033	10.9%	6.1	1151143	493452	1235	1321	5.5%	2.3
Kernel	293714	50000	321	369	1.5%	5.9	4927143	1413095	3551	4783	23.5%	3.5
TCP	178571	55952	414	12	0.9%	3.2	6304286	1966667	5345	5485	30.0%	3.2
Copies	0	0	0	60	0.0%		5525524	487500	6313	32057	26.3%	11.3
Driver	0	0	0	0	0.0%		2335810	711905	2378	5521	11.1%	3.3
Total Non-idle	9433524	3249405	16152	4152			20243905	5072619	18821	49167		

Table 2. Comparative analysis of AMP over SMP

64KB W	Improvements in		
	Cycles	L2 Misses	Clears
idle			
others	0.1%	0.8%	-1.5%
SCSI	0.8%	7.9%	-0.1%
iSCSI	-0.4%	0.7%	-1.4%
Interface	3.2%	5.9%	6.3%
Kernel	10.2%	12.3%	14.4%
TCP	11.0%	15.6%	15.9%
Copies	9.4%	8.8%	9.4%
Driver	0.2%	6.2%	3.6%
TOTAL	**34.6%**	**58.2%**	**46.8%**

64KB R	Improvements in		
	Cycles	L2 Misses	Clears
idle			
others	1.3%	0.6%	1.6%
SCSI	2.7%	10.4%	2.6%
iSCSI	0.6%	1.5%	0.2%
Interface	4.9%	7.7%	7.6%
Kernel	10.5%	12.0%	16.3%
TCP	12.1%	16.1%	20.1%
Copies	5.1%	9.9%	6.4%
Driver	2.1%	1.9%	4.7%
TOTAL	**39.3%**	**60.1%**	**59.6%**

6 Related Work

The major overheads of a software TCP stack are well documented [5, 6, 10]. Much of these overheads lie in memory touching operations that do not scale well with processor speeds. More commonly overlooked are the overheads incurred by scheduling and interrupting [10]. To improve cache locality, static affinity [7] and improved scheduling affinity in Linux-2.6 [18] are efforts to direct interrupts and processes to the most appropriate processors. While this works for relatively homogenous applications, dynamically changing applications still pose challenges. As such, other researchers have built prototypes based on functional partitioning [13, 14,15]. An interesting use of functional partitioning [3] has shielded CPUs service only user-defined high priority tasks and interrupts, to deliver real time response using standard Linux. Early works on dedicating a CPU to do specific functions were done primarily on proprietary OS on supercomputers. Processors are dedicated to perform message-passing among nodes [9, 14]. More recently, the AsyMOS project [13] logically attaches general purpose CPUs for the purpose of adding intelligence to devices. They made use of a lightweight device kernel on their device CPU, in place of a full OS. The TCP Servers project [15] defines a generic architecture to offload TCP processing to either processors or nodes. Unlike our focus on removing OS and device intrusions, their focus is the use of a kernel thread dispatcher to dynamically schedule TCP functions onto dedicated processing nodes. Communication between CPUs in [13, 15] is based on inter-processor and/or remote procedure calls, whereas we have implemented our interfaces with queues in cache-coherent shared memory.

7 Conclusion

The ability to process iSCSI at line rate is the key to enabling IP-based storage area networks. Our software implementation, not only achieved line rate performance, but also had the same efficiency as HBAs discussed in [1, 17]. We found that successful, software-based iSCSI architecture must include the following: Zero copy, or exploit thread-level parallelism to hide copy costs; Re-architect applications to leverage well-designed interfaces; and reduce/eliminate device interrupts and OS intrusions.

Working with iSCSI allowed us to gain insights to real usage models to drive API requirements. Due to the clean separation of the TCP engine, lessons learnt here are also equally applicable to offload adapters. We believe that the hard affinity approach gives the upper bound on network performance possible on existing processors and platform architectures. Because of partitioning, we were able to fully control the scheduling of protocol processing. This however does introduce resource balancing issues, where the question of PPEs to host processors ratio must be addressed. The next challenge is to architect an approach that allows for the existence of multiple partitioned PPEs, and be able to optimally balance these resources within the confines of mainstream operating systems. Looking ahead to chip-multiprocessing, where multiple cores reside on each processors, asymmetric partitioning of work to cores provides an effective and viable OS model to multiprocessing.

Acknowledgement

We would like to thank Don Cameron for being the biggest fan of our iSCSI over ETA work; and Jason Fung for his help in consolidating megabytes of data, and carefully abstracting them to a point where sane analysis can proceed.

References

1. S. Aiken, D. Grunwald and P. Andrew. A performance analysis of the iSCSI protocol. In *Proceedings of the 20th IEEE Conf. on Mass Storage Systems and Technologies*, 2003.
2. V. Anand and B. Hartner. TCPIP Network Stack Performance in Linux Kernel 2.4 and 2.5. In *Proc. of the Ottawa Linux Symposium,* Ottawa, 2002
3. S. Brosky and S. Rotolo. “Shielded Processors: Guaranteeing Sub-millisecond Response in Standard Linux”, 4th Real-Time Linux Workshop, Boston, Dec 2002.
4. D. Cameron and G. Regnier. *The Virtual Interface Architecture*. Intel Press, 2002.
5. J. Chase, G. Gallatin, K. Yocum. End system optimizations J. for high-speed TCP. *IEEE Comms, Special Issue on high-speed TCP*, 2001, 39(4).
6. Foong, T. Huff, H. Hum, J. Patwardhan and G. Regnier. TCP performance re-visited. In *Proc of the IEEE ISPASS,* Austin, Mar 2003.
7. Foong, J. Fung, D. Newell, P. Irelan, A. Lopez-Estrada, S. Abraham. Architectural characterization of the impact of Processor Affinity on Network Processing, *to appear IEEE ISPASS*, Mar 2005.
8. L. Hammond, B. Nayfeh and K. Olukotun. A single-chip multiprocessor. *IEEE Computer*, Sept 1997.
9. J. Hsu and P. Banerjee,. A message passing coprocessor for distributed memory multicomputers, In *Supercomputing*, 720 – 729, 1990.
10. J. Kay and J. Pasquale. The Importance of Non-Data Touching Processing Overheads in TCP/IP. In *Proc of ACM SIGCOMM*, 1993.
11. P. Leroux. Building Scalable Networking Equipment Using SMP. *Dedicated Systems Magazine*, 2001.
12. J. Mogul. TCP offload is a dumb idea whose time has come. In *Proc of HotOS IX*, Lihue, May, 2003.
13. S. Muir and J. Smith. AsyMOS: An asymmetric multiprocessor OS. In *Proceedings of OPENARCH '98*, April 1998.

14. P. Pierce and G. Regnier. The Paragon implementation of the NX message passing interface. In *Proc of SHPCC 94*, 1994.
15. M. Rangarajan, A. Bohra, K. Banerjee, E. Carrera, and R. Bianchini. TCP servers: Offloading TCP processing in internet servers. Technical report, Rutgers University, 2002.
16. G. Regnier, D. Minturn, G. McAlpine, V. Saletore and A. Foong. ETA: Experience with an Intel Xeon Processor as a packet processing engine. *IEEE Micro*, Jan 2004.
17. P. Sarkar, S. Uttamchandani, and K. Voruganti. Storage over IP: When does hardware support help ? In *Proc of the 2nd USENIX Conf on File and Storage Technologies*, 2002.
18. M. Meredith, D. Vianney. Linux 2.6 Performance in the Data Center. Linux World Expo, Jan 2004.
19. *Iometer performance Analysis Tool*. http://www.iometer.org.
20. Oprofile: A system-wide profiling tool for Linux. http://oprofile.sourceforge.net.
21. *UNH-iSCSI, Intel iSCSI reference stacks*. http://sourceforge.net/.
22. Scalable Networking: Eliminating the Receive Processing Bottleneck—Introducing RSS. http://www.microsoft.com/whdc/
23. Remote direct data placement protocol. http://www.ietf.org/html.charters/rddp-charter.html

Reorder Density (RD): A Formal, Comprehensive Metric for Packet Reordering[1]

Nischal M. Piratla, Anura P. Jayasumana, and Abhijit A. Bare

Department of Electrical and Computer Engineering, Colorado State University,
Fort Collins, Colorado 80523, USA
{Nischal.Piratla, Anura.Jayasuamna, Abhijit.Bare}@colostate.edu

The increase in link speeds, increased parallelism within routers and switches, QoS support and load balancing among links, all point to future networks with increased packet reordering. Unchecked, packet reordering will have a significant detrimental effect on the end-to-end performance, while resources required for dealing with packet reordering at routers and end-nodes will grow considerably. A formal analysis of packet reordering is carried out and Reorder Density (RD) metric is defined for measurement and characterization of packet reordering. RD captures the amount and degree of reordering, and can be used to define the reorder response of networks under stationary conditions. Properties of RD are derived, and it is shown that the reorder response of the network formed by cascading two subnets is equal to the convolution of the reorder responses of individual subnets. Packet reordering over the Internet is measured and used to validate the derivations.

1 Introduction

The reasons for out of order arrival of packets include but are not limited to: (i) packet striping at layer 2 and 3 links, i.e., when an earlier packet is placed in a longer queue and later packet in a shorter queue, the packets may arrive out of order [4,7], (ii) retransmissions on wireless links [3] and due to TCP, (iii) diffServ scheduling where the flow that exceeds the constraints, e.g., the non-conformant packets are dropped or given a lower priority leading to the packet placement in different queues resulting in an out-of-order delivery [6], and (iv) route fluttering where for example a route may oscillate due to dynamic load splitting among the links. In such cases, different packets of the same stream take different routes leading to different delays [12].

Packet reordering, irrespective of its cause, impacts applications based on both TCP and UDP significantly. In the case of TCP, when packets in forward path go out of order, the receiver may perceive packets as lost, resulting in a reduced congestion window, and increased number of retransmissions [4,5] that further degrade the performance. Reverse-path reordering, i.e., reordering of acknowledgements, results in the loss of TCP's self-clocking property, leading to bursty transmissions and

[1] This research was supported in part by NSF ITR Grant No. 0121546, Ixia University Partners Program, and Agilent Technologies.

R. Boutaba et al.(Eds.): NETWORKING 2005, LNCS 3462, pp. 78 – 89, 2005.

possibly to increased congestion [4]. Approaches for mitigating the impact of out-of-order packet delivery on TCP performance include adjusting ‘dupthresh’ parameter, i.e., the number of duplicate ACKs to be allowed before classifying a following non-acknowledged packet as lost [19]. In delay sensitive applications based on UDP, e.g., IP telephony, an out-of-order packet that arrives after the elapse of playback time is treated as lost thereby decreasing the perceived quality of voice. To recover from reordering, the out-of-sequence packets are buffered until they can be played back in sequence to the application. Thus, an increase in out-of-order delivery by the network consumes more resources at the end-hosts, and also affects the end-to-end performance of the applications.

There is an effort to address this issue at intermediate nodes, at IP level. Many contemporary routers attempt to eliminate the reordering caused by the scheduling schemes within these nodes by either a) input reordering, i.e., identifying the individual streams and forwarding the packets of same stream to the same queue thus preventing reordering, or b) output reordering, i.e., buffering packets at the output of the router to ensure that the packets belonging to the same stream are released in order of their entry into the node [9]. For example, the network processors from vendors such as IBM, Motorola, Vitesse, TI and Intel, have built-in hardware to track flows. While these approaches reduce the reordering that occurs inside a router, they cannot eliminate reordering due to multiple paths. Furthermore, the complexity of these approaches will increase significantly as the number of parallel flows in a pipe increases (due to the need to keep information on a large number of parallel flows), and as the ratio of packet time to routing latency decreases.

The delay in sequencing the packets back in order (as in output buffering) is proportional to log (C_s/U) where C_s is the capacity of the link and U the packet size [18]. The buffer size requirements to put the packets back in order are also shown to increase dramatically with link speed. With the increase in number of flows, keeping track of them becomes difficult too. Moreover, the number of table entries in routers are growing exponentially [17], which means that in spite of high-speed links in future, packets will spend more time within the network, i.e., a higher end-to-end delay to packet time ratio, leading to more load splits and higher packet reordering. Increase in latency required at the intermediate switches and routers to reorder the packets add to the end-to-end latency and the RTT, thereby affecting the performance as well [2].

Next generations of networks have to deal with the problem of out-of-sequence packets proactively. However, scant attention has been paid so far towards understanding the nature of reordering that occurs in networks. Characteristics of reordering that occurs in a given end-to-end path may be used to enhance the ways that the protocols deal with this problem. Measurement and characterization of reordering in packet sequences and models that provide insight into this problem can lead to the development of scalable techniques to deal with reordering and tools that diagnose network problems.

This paper investigates reordering of packets and proposes a metric, Reorder Density (RD), for measuring reordering in a packet sequence. This metric captures the magnitude and statistical properties of reordering occurring in a network. Thus, we define the reorder response of a network, as the RD of the sequence leaving the network corresponding to an in-order input packet sequence. Further, it is shown that the reorder response of a cascaded pair of networks corresponds to the convolution of the reorder responses of the individual networks. This allows us to systematically measure, model and evaluate reordering in complex networks. Next section identifies the requirements of metrics for packet reordering that will make the metric a comprehensive and a useful tool to capture, model, and study packet reordering. In Section 3, the reorder problem is formulated and the proposed metric RD is presented. Section 4 presents measurements of reordering over the Internet for a range of network spans, and verifies the model for computing reordering on cascaded networks. Conclusions are presented in section 5.

2 Metrics for Reordering

The percentage of out-of-order packets has been used as a metric for characterizing packet reordering [3,6]. This method is vague, incomplete and does not provide information about the nature of reordering. Consider two packet sequences (1,3,4,2,5) and (1,4,3,2,5). It is possible to interpret the out-of-orderliness of packets differently in these cases, for example, packets 2, 3 and 4 are out-of-order in both the cases or only packet 2 is out-of-order in the first sequence, while packets 2 and 3 are out-of-order in second case or packets 3 and 4 are out-of-order in both the cases. Even with a convention such as only late packets are considered reordered, the measure is insufficient [15]. Percentage does not capture the amount by which a packet is out of order. The amount of displacement of a packet directly influences the complexity of hardware/software to recover from the reordering. In the case of a sequence such as (1,3,4,2,5), if buffers are available to store packets 3 and 4 while waiting for packet 2, it is possible to recover from the reordering. However, there may be cases where the application requirement is such that arrival of packet 2 after this delay renders it useless. For such cases, we could place a threshold that treats any packet that is more than 2 places late as lost. While one can argue that a good packet reordering measure should capture such effects, a counter argument can also be made that packet reordering should be measured strictly with respect to the order of delivery and should be application independent. A framework for metrics presented in [13] states that "The metrics must be useful to users and providers in understanding the performance they experience or provide."

A metric for capturing out-of-order nature of a packet sequence ideally should have the following properties:

1. Simplicity: The measure should be simple, yet contain enough information to be useful.
2. Orthogonality: Metric should, to the extent possible, be independent or orthogonal to other phenomena that affect the packet streams, e.g., packet loss and duplication.
3. Differentiability: Metric should provide insight into the nature of reordering, and perhaps even into possible causes. It should capture both the amount and extent of reordering.
4. Usefulness: Rather than being a mere representation of the amount of reordering in a packet stream, reorder metric must be useful to the application and/or

resource management schemes. For example, it may allow one to determine the size of buffer that is required to recover from reordering.

5. Evaluation complexity: The metric should be computable in real-time. In evaluating reordering in an arbitrarily long sequence, one should be able to keep a running measurement, without having to wait till all the packets have arrived. The memory requirement, i.e., the amount of state information, should not grow with the length of the sequence (N) and the computation time should be O (N).
6. Robustness: Reorder measurement should be robust against different network phenomena and measurement peculiarities such as a very late arrival of a duplicate packet or a burst of losses.
7. Broader Applicability: A good metric would have applicability beyond just characterizing the nature of reordering in a given sequence of packets. For example, a good metric may allow one to combine the characteristics of individual networks to predict the reorder behavior of the cascade of these networks. Regeneration of a sequence that follows the measure is also a very useful application.

Recently, Internet Engineering Task Force (IETF) has presented a few metrics for reordering [8,11]. The metrics in [11], a work in progress, are classified into sample metrics and receiver assessment metrics. Sample metrics are defined on a sample of sequence numbers and percentage reordering can be an example of such a metric. However, these metrics fail to meet many of the criteria mentioned above, especially those related to differentiability, usefulness and robustness. The receiver assessment metric mainly relies on n-reordering, which looks only at few late packets. For sequence (1,4,5,2,3,2); the n-reordering method treats 1, 4 and 5 as 0-reordered. Since 2 is late by 2 places, it is 2-reordered but 3 is 0-reordered whereas duplicate 2 is 1-reordered. Besides the ambiguity in the definition, this metric is not orthogonal to duplication. In addition, the metric uses a buffer to store current arrivals to compute the n-reordering of the future arrivals leading to larger buffer requirements for higher amount of reordering. The reorder metric proposed in [1] measures the occupancy density of the reorder buffer, and as such will be referred to as Reorder Buffer-occupancy Density (RBD) [8].

3 Analysis of Reordered Sequences

In this section, we develop a general formulation for characterizing packet reordering, and define Reorder Density (RD). We then evaluate the reorder response of a cascade of networks in terms of the reorder responses of the individual subnets.

3.1 Representation of Reordering

Without loss of generality, consider a sequence of packets (1,2…N) transmitted over a network. A receive_index (1,2…) is assigned to each packet as it arrives at the destination. Lost and duplicate packets are not assigned a receive_index. First consider the case in which no losses or duplication of packets occur in the network. If the receive_index assigned to packet m is (m + d_m), with $d_m \neq 0$, we say that a reorder event has occurred, and this event is denoted by r(m, d_m) . In the absence of reordering the sequence number of the packet and the receive_index are same, i.e., $d_m = 0$ for each packet. A packet is late if d_m > 0, and early if d_m < 0. Thus, packet reordering of a sequence of packets is completely represented by the union of reorder events, R, referred to as the reorder set: $\text{R} = \bigcup_m \{\text{r(m},d_m) \mid d_m \neq 0\}$

Table 1. (a), (b) and (c) Examples of reordered sequences with corresponding R

Arrived Sequence	1	2	3	5	4	6	8	7
Receive_index	1	2	3	4	5	6	7	8
Displacement	0	0	0	-1	1	0	-1	1

(a) No losses or duplicates R = {(4,1), (5,-1), (7,1), (8,-1)}

Arrived Sequence	1	2	5	3	6	7	8	9
Receive_index	1	2	3	5	6	7	8	9
Displacement	0	0	-2	2	0	0	0	0

(b) Packet 4 is lost R = {(3, 2), (5, -2)}

Arrived Sequence	1	2	6	4	3	5	3	3
Receive_index	1	2	3	4	5	6	-	-
Displacement	0	0	-3	0	2	1	-	-

(c) Packet 3 is duplicated R = {(3, 2), (5, 1), (6, -3)}

If there is no reordering in a packet sequence then R = {ϕ}. Conventionally, we represent R with non-decreasing order of m. Table 1(a)-(c) show the examples of the arrived sequence (sequence number), assigned receive_index and displacement, as well as the corresponding reorder sets for three cases: a) without losses or duplication, b) with loss, and c) with duplication of packets.

Lemma 1: In the absence of losses and duplicates, the sum of displacements of all packets in a sequence is equal to zero, i.e. $\sum d_i = 0$.

Proof: Consider a packet sequence of size N: (1,2 …N). If d_i *denotes the displacement of* i^{th} *packet due to reordering, then the new positions of the packets 1,.. N are:*

$$(1+d_1),\ (2+d_2),\ ..,\ (i+d_i)..,\ (N+d_N)$$

Since the number of positions at the sender and receiver are equal, the sum of receive_indices is the sum of integers from 1 to N. Therefore we have,

$$\sum_{i=1}^{N} i = \sum_{i=1}^{N} (i+d_i) \Rightarrow \sum_{i=1}^{N} d_i \quad \text{(Q.E.D.)}$$

Next, consider the case where packets may be lost or duplicated in transit. Assume that the loss of a packet can be detected at the receiver. We skip the receive_index corresponding to the sequence number of the lost packet, i.e., if packet ‘e’ is lost, then the receive_index = e is not assigned. In case of duplicates, we consider only the first copy of the packet at the receiver end, and discard the duplicate, i.e., the duplicate is not assigned a receive_index. These two cases are illustrated in Tables 1 (b) and (c) where the packet with sequence number 4 is lost and the packet with sequence number 3 is duplicated respectively. How to detect the loss of packets on the fly to skip the receive_index and deal with duplicate packets in a measurement environment, is addressed in subsection 3.4 below and [8] in detail.

Lemma 2: When lost and duplicate packets are detected and receive_index is assigned as specified, Lemma 1 holds true, i.e., $\sum d_i = 0$.

Proof: Consider the case where packet ‘ e ’ is lost and ‘ j ’ is duplicated. Since we ignore the duplicate copy of j, no index is assigned to the duplicate of that packet. As the index ‘ e ’ is reserved for the lost packet, it is not used in the (N-1) receive indices. Thus, the sum of (N-1) receive_index values assigned at the receiver is

$$\sum_{i=1, i \neq e}^{N} (i + d_i) = \sum_{i=1}^{N} (i) - e \Rightarrow \sum d_i = 0 \text{ (Q.E.D).}$$

3.2 Reorder Density (RD)

RD is defined as the discrete density of the frequency of packets with respect to their displacements, i.e., the lateness and earliness from the original position. Let $S[k]$ denote the set of reorder events in R with displacement equal to k, i.e.,

$$S[k] = \{\mathrm{r}(\mathrm{m}, d_m) \mid d_m = k\} \ .$$

Let $|S[k]|$ be the cardinality of set $S[k]$. Thus, RD[k] is defined as $|S[k]|$ normalized with respect to the total number of received packets (N'). Note that N' does not include duplicates or lost packets. RD[0] corresponds to the packets for which receive index is the same as the sequence number. Thus

$$\mathrm{RD}[\ k\ ] = |S[k]| / \mathrm{N'} \text{ for } k \neq 0 \text{ and } \mathrm{RD}[0] = 1 - \sum_{k \neq 0} |S[k]| / \mathrm{N'}$$

A reorder histogram function, where $|S[k]|$ is used directly without normalization, may prove useful for certain applications, where the actual number of packets that is reordered is given. The results presented in this paper can easily be modified to cover such a case.

Lemma 3: Reorder Late/Early density function (RD) satisfies $\sum_k (k * \mathrm{RD}[k]) = 0$

Proof: Since RD [k] implies that there are $N' * RD[k]$ *packets with displacement k, the sum of displacements of packets is given by*

$$\sum_k (k * \mathrm{N'} * \mathrm{RD}[k]) = \mathrm{N'} * \sum_k (k * \mathrm{RD}[k]) = \sum_m d_m = 0 \textit{ (from Lemma 2) (Q.E.D)}$$

3.3 Reorder Response of Networks

The characterization of reordering in a sequence of packets in the form of RD, in addition to capturing the out-of-orderliness of a packet sequence accurately, also allows us to model networks with respect to reordering introduced by sub-networks forming the network.

First note that RD corresponding to a sequence of in-order packets is a unit impulse (at $k = 0$). Consider sending this sequence of packets through a network. RD of the sequence observed at the output corresponds to the reordering introduced by the network, and hence we call it the reorder response ($\boldsymbol{J}[k]$) of the network. Note that ($\boldsymbol{J}[k]$) is defined with respect to an input and an output of a network. Thus each input/output port pair of the network has a corresponding reorder response. $\boldsymbol{J}[k]$ of a network in which there is no reordering corresponds to a unit impulse at k=0. $\boldsymbol{J}[k]$ can also be interpreted as the probability that a certain packet gets displaced by k.

The reorder response of a network captures the effect of the network on a sequence of packets flowing from the source port to the destination port of interest. Since a network is a time-varying and highly dynamic environment, it may not be possible to characterize the network completely using a single reorder response function. In fact, the reorder response will be a function of many factors including the background or cross traffic in the network, and statistical characteristics of the inter-packet gap. We hypothesize however that a reorder response exists for a network operating under stationary conditions and a given inter-packet gap distribution, provided the reordering introduced by the network is not dependent on the sequence number of the packet. This excludes, for example, a node that looks at the packet sequence number and puts the packets in a certain order. In fact, a set of measurements that we have carried out supports this hypothesis [15]. Thus the following discussion applies only to networks operating under stationary conditions, i.e., statistical characteristics of the network and packet sequence do not change with time. This excludes, for example, a node that looks at the packet sequence number and puts the packets in a certain order. We expect that a theory developed for reordering in networks under such stationary conditions will lead the way towards a more general solution in future.

Theorem: The reorder response ***J[k]*** *of a network formed by cascading two subnets, with reorder responses* ***J₁[k]*** *and* ***J₂[k]*** *respectively, is given by the convolution of* ***J₁[k]*** *and* ***J₂[k]*** *, i.e.,* $\boldsymbol{J}[k] = \boldsymbol{J}_1[k] * \boldsymbol{J}_2[k]$.

Proof: Consider an arbitrary packet with sequence number 'm' that enters the cascade of two networks. It undergoes a displacement $d_m^{(1)}$ *in Net-1, corresponding to the event* $\mathrm{r}(\mathrm{m}, d_m^{(1)})$. *Thus, packet m occupies the position of m' as it enters Net-2, with* $m' = m + d_m^{(1)}$. *When the packet with sequence number m' enters Net-2, it undergoes a displacement* $d_m^{(2)}$ *in Net-2. The overall displacement of the packet* $d_m = d_m^{(1)} + d_m^{(2)}$,

The corresponding reorder event is $\mathrm{r}(\mathrm{m}, d_m) = \mathrm{r}(\mathrm{m}, d_m^{(1)} + d_m^{(2)})$.

The probability distribution of $d_m^{(1)}$ *and* $d_m^{(2)}$ *are given by reorder responses of Net-1* ***J₁[k]****, and Net-2* ***J₂[k]****, respectively. Since* $d_m = d_m^{(1)} + d_m^{(2)}$ *, and the packet* *reordering is considered to be independent of the sequence number, the probability density distribution of* d_m *is given by the convolution of the distributions of* $d_m^{(1)}$ *and* $d_m^{(2)}$ *, i.e.,* $\boldsymbol{J_1[k]*J_2[k]}$ *(Q.E.D)*

3.4 Evaluation of RD at the Receiver

Here we describe the evaluation of RD at the receiver. Lost packets and duplicates are taken into account by skipping the receive_index corresponding to lost packets, and not assigning a duplicate packet a receive_index. However, this process needs detection of losses and duplicates, both of which provide implementation challenges.

When can a packet be considered lost? One possibility is to consider it lost if it does not arrive when it is expected, but if it arrives later, go back and make appropriate corrections. At the other extreme, one can wait till the end of the received sequence to declare a packet as lost. However this requires keeping track of all received packets, as well as applying corrections to computations performed so far. Both these approaches are memory consuming, and also preclude real-time evaluation of the metrics. Maintaining a threshold D_T and an early arrival buffer addresses this problem. If a packet is not received within D_T packets from where it is expected, it is considered lost. A packet is classified as a duplicate on its arrival, if it already exists in early arrival buffer or current D_T window or the packet number is less than current receive_index. With the use of threshold D_T, since it is not known whether a packet is lost until D_T packets are received, real-time RD evaluation may be done in one of two ways:

1. Go-back D_T: In this method, the rules are applied at each arrival. If a packet that was supposed to arrive D_T places ago does not arrive, then this sequence number is removed from the receive_index, and RD is recomputed for the previous D_T steps. Consider a received sequence (1,3,4,5,6,7,2) and D_T = 3. As soon as 5 arrives, 2 is classified as lost and we go back and correct the previous D_T receive_indices and displacements. When 2 actually arrives later, we do not assign receive_index to this arrival, i.e., consider it as lost and discard the packet. This method requires recording the previous D_T packet numbers, and additional processing as we recomputed offsets when a packet is lost. However, if the amount of reordering is low, the overall computation is quicker than the next method.
2. Stay-back D_T: Here the computation lags by D_T packets, i.e., the packet with receive_index *i* is not used in the evaluation until D_T more packets have arrived after that. Thus, we do not correct or adjust any displacements. This method also requires buffering of the next D_T arrivals.

A small D_T value is used in above illustrations of the concept for convenience. It may be set much higher for practical measurements. A larger threshold value results in higher memory requirements for these implementations. However, the computation complexity in both cases is of the order N, where N is the size of the received sequence. Use of the threshold D_T also improves the robustness of RD. For example, if a packet was late by a large number say 1000, then the next 1000 packets would be shown as early in the absence of a threshold. By using a threshold, we can eliminate such large impacts on RD due to a single reordering event. Furthermore, it allows us to recover from conditions such as a large number of missing or duplicate packets. We have not completely described this aspect in the present paper. Perl scripts and algorithms are available in [14].

4 Measurements and Results

This section presents measurement-based results to illustrate and characterize the reordering in networks using the RD concept. To measure reordering, using RD, multiple files were downloaded to a host on subnet 129.82.x.x (in Colorado, USA) from different sites around the world. To make sure that the connection is not short-lived, sites with downloadable files of sizes more than 2MB were chosen. Using IP2Location software and ping results, we obtained the geographical location, the average one-way delay value (D) and approximate standard deviation of the delay (SD). We are presenting only 3 of 10 sets of measurements [15] here, corresponding to a range of end-to-end delays. TCP based applications namely; HTTP and FTP were used for servers in USA (209.211.x.x) and Italy (62.94.x.x) respectively. Data collected from them had D = 45.5 ms and 81.9 ms, and SD = 0.81 ms and 0.19 ms respectively. From host on subnet 130.105.x.x, a media file was played and packets had D = 95.4 ms and SD = 0.17 ms. Using 'tcpdump' tool to collect the information of received data, the TCP and RTP sequence numbers were mapped to (1,2 …). Fig. 1 shows the observed RDs, which is also the reorder response of the network, from these sites. (Vertical axis is broken to increase the clarity of diagrams).

RD provides a comprehensive measure of reordering and we can draw a number of inferences from these measures: (i) Net-1 has the smallest average delay, but due to larger deviations, the amount of reordering is comparatively high. Conversely, looking at the shape of RD, we can comment on the delay deviations in the network. The wider the RD spread, the higher the variance in delay. (ii) For Net-1 it is evident that the network deviates from the normal expectation, as large number of packets arrives reordered. Knowing the corresponding RD, we can tune this network by allocating a larger buffer size to recover from reordering in UDP application or increasing the number of duplicate ACKs to wait before fast retransmit with TCP [10, 12]. (iii) For Net-2, the application can recover from reordering by having a buffer size equal to 2 packets. Although Net-2 has approximately 2% more reordering than Net-3, applications using Net-2 will perform better due to lower displacements of reordered packets. (iv) In the case of Net-3, the RD has a discontinuity. One percent of the packets can be as late as 4 positions. It is possible that due to phenomenon like packet stripping in a network, these packets take an alternate path. Here, instead of using triple-ACK for *fast retransmit*; we could use 2-ACK, given the effect of 1% reordering on performance is acceptable or use 4-ACK to account of all reordered packets.

To verify the convolution theorem for reordering, we emulated two networks using NISTNet on Linux boxes. The connectivity used is shown in Fig. 2. RDs were used as reorder responses for the networks. Net-X was emulated using mean (D) and standard

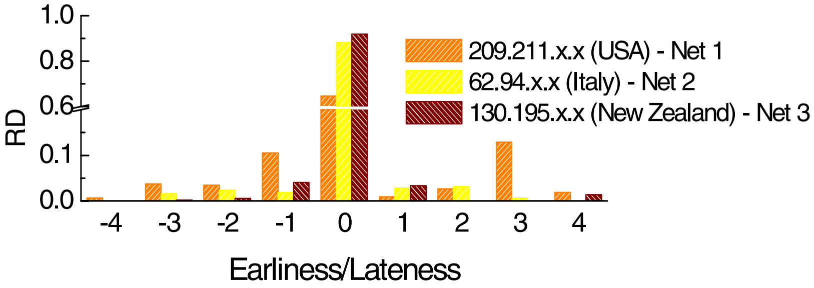


Fig 1. RD based on measurements for Net-1, 2 and 3. ($D_T = 5$)

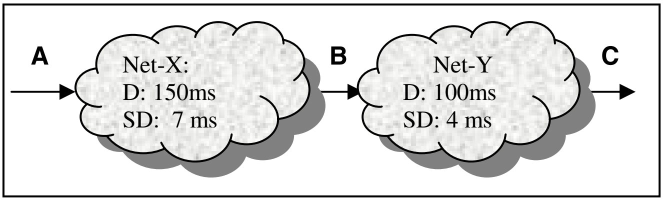


Fig 2. Emulated network topology to verify convolution theorem for RD

deviation (SD) of delays as 150 ms and 7 ms respectively and Net-Y is emulated using D = 100 ms and SD = 4 ms. Packet streams of sizes 10,000 packets were sent from A to B and B to C separately. Another packet stream of 10,000 packets was sent from A to C on this cascade. Constant inter-packet gap (of 500µs) was maintained in each of the input streams. These emulators running on the workstations were then connected using a cross-cable. The RD computed in this case was compared to the result based on convolution of the RDs for A to B and B to C cases. This comparison is shown in Fig. 3. The mean square error (MSE) between the two was equal to 8.77e-06, strongly validating the theorem.

It is interesting to note that for the packet stream from A to C, even though the inter-packet gap at A was 500µs, by the time packets transit through B, the inter-packet gap is no longer a constant. On the other hand, the reorder response for Net-Y used for convolution, shown in Fig. 3(a) corresponds to a constant inter-packet gap. This indicates that the reorder response of a network, while dependent on the inter-packet gap distribution in general, may not be highly sensitive to it. This is an aspect of reorder response that we continue to investigate.

5 Conclusions

The existing metrics to measure reordering are vague and insufficient to characterize packet reordering. We have presented a formal method for representing out-of-order sequences of packets, and defined the reorder density (RD) metric. RD characterizes

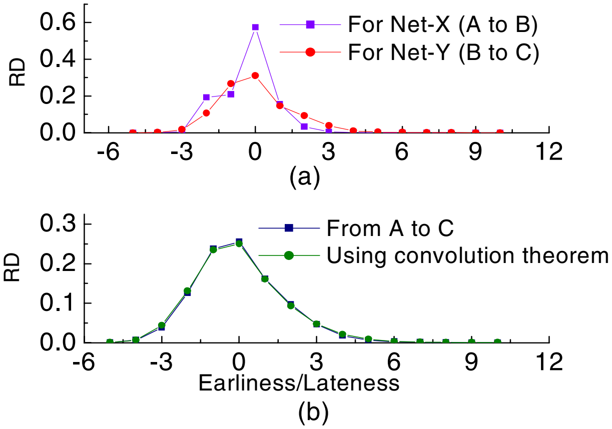


Fig. 3. (a) RD from A to B and B to C with separate streams (b) Verification of the theorem of convolution using network emulators for RDs as reorder responses

and measures packet reordering comprehensively, is simple, and orthogonal to losses and duplicates. It captures both the amounts of packets affected and the magnitude of reordering, and can be used as the probability density function corresponding to the displacement of an arbitrary packet. The metric can be evaluated in real-time as the packets arrive at a node. A threshold D_T limits the complexity of implementation, by considering a packet that is late by D_T to be lost. The computation complexity of the algorithm is O(N), where N is the number of packets in the sequence. The memory requirement for the implementation is proportional to D_T. The use of D_T also makes it robust, allowing it to recover from cases such as very early or very late packets, and sequences of duplicates, or bursty losses. Further, the metric can be used to characterize the reordering introduced by a network, and under a fairly broad set of conditions, the reorder measurement of different subnets can be combined to predict the end-to-end reorder characteristics of a network. Currently, sequence regeneration algorithm is available for RD measures [2].

Reorder response of a network depends on factors such as the network load, background traffic, and the distribution of the inter-packet gap. At high sending rates, inter-packet gaps have negligible correlations, also validating convolution results [16]. Our present work includes measurements to understand packet reordering over the Internet in more detail. By keeping track of RD for an on-going connection, one can dynamically tune transport protocols to obtain superior performance.

References

1. Banka, T., Bare, A. and Jayasumana, A., “Metrics for Degree of Reordering in Packet Sequences,” Proc. of the IEEE 27th LCN, Nov. 2001, pp. 333-342.
2. Bare, A. A., “Measurement and Analysis of Packet Reordering,” Masters Thesis, Department of Computer Science, Colorado State University, 2004.

3. Bellardo, J. and Savage, S., “Measuring Packet Reordering,” Proc. of Internet Measurements Workshop (IMW’02), Nov. 2002, pp.97-105.
4. Bennett, J. C. R., Partridge, C. and Shectman, N., “Packet Reordering is Not Pathological Network Behavior,” Trans. on Networking IEEE/ACM, Dec. 1999, pp.789-798.
5. Blanton, E. and Allman, M., “On Making TCP More Robust to Packet Reordering”, ACM Computer Comm. Review, 32(1), Jan. 2002, pp.20-30.
6. Bohacek, S., Hespanha, J., Lee, J., Lim, C. and Obraczka, K., “TCP-PR: TCP for Persistent Packet Reordering,” Proc. of the IEEE 23rd ICDCS, May 2003, pp.222-231.
7. Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J. and Towsley, D., “Measurement and Classification of Out-of-sequence Packets in Tier-1 IP Backbone,” Proc. of IEEE INFOCOM, Mar. 2003, pp. 1199- 1209.
8. Jayasumana, A., Piratla, N. M., Bare, A. A., Banka, T., Whitner R., and McCollom, J., “Reorder Density Function - A Metric for Packet Reordering Measurement,” IETF draft (work in progress).
9. Liu, H., “A Trace Driven Study of Packet Level Parallelism,” Proc. of International Conference on Communications (ICC’02), New York, NY, 2002, pp. 2191-2195.
10. Loguinov, D. and Radha, H., “End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis,” Proc. of IEEE INFOCOM, Jun. 2002, pp.723-732.
11. Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and Perser, J., “Packet Reordering Metric for IPPM,” IETF draft (work in progress).
12. Paxson, V., “Measurements and Analysis of End-to-End Internet Dynamics,” Ph.D. Dissertation, Computer Science Department, University of California, Berkeley, 1997.
13. Paxson, V., Almes, G., Mahdavi, J. and Mathis, M., “Framework for IP performance metrics,” RFC 2330.
14. Perl scripts for RD, http://www.cnrl.colostate.edu/Reorder_perl_scripts.html. Last modified on Nov. 3, 2004.
15. Piratla, N., “Metrics, Measurements and Techniques for the End-to-end Characterization of Networks (tentative),” Ph. D. Dissertation, Department of Electrical and Computer Engineering, Colorado State University, (work in progress).
16. Piratla, N. M., Jayasumana A. P., and Smith H., "Overcoming the Effects of Correlation in Delay Measurements using Inter-Packet Gaps," Proc. of IEEE International Conference on Networks (ICON), Singapore, Nov 2004, pp. 233-238.
17. Ruiz-Sanchez, M., Biersack, E. W. and Dabbous, W., “Survey and Taxonomy of IP Address Lookup Algorithms,” IEEE Network, Mar./Apr. 2001, pp. 8-23.
18. Xia, Y. and Tse, D., “Analysis on Packet Resequencing for Reliable Network Protocols,” Proc. of IEEE INFOCOM, San Francisco, CA, Mar. 2003, pp. 990-1000.
19. Zhang, M., Karp, B., Floyd, S., and Peterson, L., “RR-TCP: A Reordering-Robust TCP with DSACK,” Proc. The Eleventh IEEE International Conference on Networking Protocols (ICNP 2003), Atlanta, GA, Nov. 2003, pp. 95-106.

Batch Scheduling Algorithms for Optical Burst Switching Networks

Ayman Kaheel and Hussein Alnuweiri

University of British Columbia, Vancouver BC V6T 1Z4, Canada
{aymank, hussein}@ece.ubc.ca

Abstract. Previously proposed wavelength scheduling algorithms in optical burst switching networks process each reservation request individually and in a greedy manner. In this paper we propose a new family of wavelength scheduling algorithms that process a batch of reservation requests together instead of processing them one by one. When a control burst with a reservation request arrives to a free batch scheduler, the scheduler waits for a small amount of time, called the *acceptance delay*, before deciding to accept or reject the reservation request. After the acceptance delay has passed, the scheduler processes all the reservation requests that have arrived during the acceptance delay, then it accepts the requests that will maximize the utilization of the wavelength channels. We describe an optimal batch scheduler that serves as an upper bound on the performance of batch scheduling algorithms. Furthermore, we introduce two heuristic batch scheduling algorithms. The performance of the proposed algorithms is evaluated using a discrete-event simulation model. Simulation results suggest that batch schedulers could decrease the blocking probability by 25% compared to the best previously known wavelength scheduling algorithm.

1 Introduction

Optical Burst Switching (OBS) [1] has been proposed as an optical switching technique that combines the advantages of both Wavelength-Routed (WR) networks and optical packet switching networks. As in WR networks, there is no need for buffering and electronic processing for data at intermediate nodes. At the same time, OBS increases the network utilization by reserving the optical channel for a limited time period. The basic switching entity in OBS is a burst. A burst is a train of packets moving together from one ingress node to one egress node and switched together at intermediate nodes. An optical burst consists of two parts, a header and a data burst. The header is called the control burst (CB) and is transmitted separately from the data, which is called the data burst (DB). The CB is transmitted first on a separate signaling channel to reserve the bandwidth along the path for the corresponding DB. After a given delay, the CB is followed by the DB, which travels over the same path reserved by the CB. The delay between sending the CB and the DB is called the *burst offset time*. The value of the offset time is chosen to be greater than or equal to the total

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 90–101, 2005.

processing delay encountered by the CB. Consequently, no buffering is needed for the DB at intermediate nodes.

Several signaling protocols have been proposed for OBS [1,2]. Qiao et. al have proposed a protocol called Just-Enough-Time (JET) [1]. The JET protocol is reserve-a-fixed duration scheme that reserves resources exactly for the transmission time of the burst. In JET, the CB contains the destination address, the DB length, the wavelength on which the associated DB will arrive, and the burst offset time. When a CB arrives at a core node, a wavelength scheduling algorithm is invoked to find a suitable wavelength channel on the outgoing link for the corresponding DB. The wavelength channel is reserved for a duration equal to the burst length starting from the arrival time of the DB. The information required by the scheduler such as the burst arrival time and its duration are obtained from the CB. The scheduler keeps track of the availability of free time intervals (voids) on every wavelength channel.

The absence of the concept of "packet queues" in OBS nodes, coupled with the one way nature of the JET reservation protocol, drives the blocking probability to become the main performance measure in OBS networks. The burst blocking probability is the probability that a wavelength reservation request will not be granted due to the unavailability of a free wavelength. The blocking probability at an OBS node depends to a certain degree on how efficiently can the wavelength scheduling algorithm handle voids on wavelength channels. This fact has led to a growing interest in the area of wavelength scheduling in OBS networks. A number of wavelength scheduling algorithms have been proposed in the literature (for a survey see [3]).The main differentiating factor between previously proposed algorithms is the wavelength selection criteria. If there is more than one wavelength that can be assigned to the incoming burst, each algorithm selects a wavelength according to different criteria.

In this paper we propose a new family of wavelength schedulers. The proposed schedulers process a batch of bursts together instead of processing them one by one as in the case of previously proposed algorithms. The rest of this paper is organized as follows. Section 2 gives a brief overview of wavelength scheduling algorithms. We introduce the batch scheduling class of schedulers in Section 3. Section 4 describes the optimal batch scheduler. In Section 5 we introduce two new heuristic batch scheduling algorithms. The proposed algorithms are evaluated in Section 6 using a discrete event simulation model. We conclude the paper in Section 7.

2 Wavelength Scheduling

In OBS networks employing the JET protocol (OBS-JET networks), a CB arriving at a node represents a wavelength reservation request. The reservation request consists of a pair of values (t_a, t_e). The value t_a is determined based on the offset time, and t_e is determined based on t_a and the length of the burst b, as shown in Figure 1. A wavelength scheduling algorithm X is more efficient than wavelength scheduling algorithm Y, if it increases the utilization of the

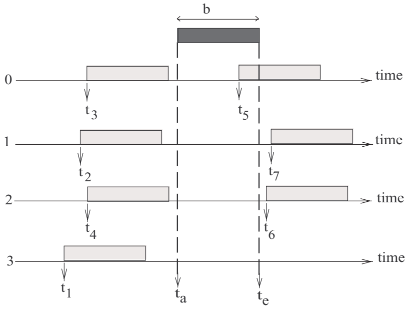


Fig. 1. Wavelength Scheduling

wavelength channels, and hence decreases the burst blocking probability. The blocking probability is calculated as the ratio of bits blocked to bits sent.

A wavelength channel is said to be unscheduled at time t when no burst is using the channel at or after time t. A channel is said to be unused for the duration of voids between successive bursts and after the last burst assigned to the channel.

Xiong *et* al. [4] proposed a channel algorithm called *LAUC-VF* (Latest Available Unused Channel with Void Filling). The basic idea of the LAUC-VF algorithm is to minimize voids by selecting the latest available unused data channel for each arriving DB. Given the arrival time t_a of a DB with duration b to the optical switch, the scheduler first finds the outgoing data channels that are available for the time period of $(t_a;\ t_a + b)$. If there is at least one such data channel, the scheduler selects the latest available data channel, i.e., the channel having the smallest gap between t_a and the end of the last DB just before t_a. The LAUC-VF algorithm is widely considered to be among the best wavelength scheduling algorithms in terms of its blocking probability.

3 Batch Scheduling

Previously proposed wavelength schedulers are considered to be greedy algorithms. They are greedy in the sense that they consider every request individually, and make the choice that looks best at the moment. Therefore, if the request can be accepted (unblocked), it will be indeed accepted. When a CB arrive to an OBS node employing a greedy wavelength scheduling algorithm, it is inserted into a FIFO queue. When the wavelength scheduler is free, it dequeues the first packet on the queue and processes it. Figure 2 describes this process.

In this work we pose the following question: What if we defer the acceptance of some unblocked requests until we see more requests? The intuition behind this question is that it may be advisable to reject an initially unblocked request if later requests will make better use of the wavelength channels.

To further illustrate the idea. Let the acceptance delay be d. Consider the case of $d = \infty$. In this case the system will wait until all reservation requests

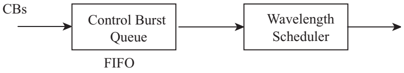


Fig. 2. Control burst scheduling

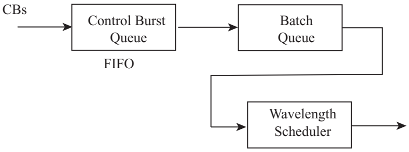


Fig. 3. Control burst scheduling with batch queue

have been received, then it will select requests that will maximize the utilization. Obviously the case of $d = \infty$ is impractical. Our proposal is to use an acceptable value of d, such that we schedule a batch of reservation requests together, instead of using $d = 0$ as in the case of greedy algorithms.

To implement the above concept, we modify the burst scheduling process by inserting a queue before the wavelength scheduler and after the CB queue, we call it the *batch queue*, as shown in Figure 3. When a CB arrives to an OBS node, it gets inserted into the CB queue. When the wavelength scheduler becomes free, it will wait for a small amount of delay, viz. the *acceptance delay* d, then it moves all bursts that are in the CB queue to the batch queue, then processes all the CBs in the batch queue together instead of the previously used greedy approach.

The benefit of the batch queue, as will be seen later, is to limit the delay that a CB can incur during the batch processing.

3.1 Definitions

A graph is $G = (V, E)$ consists of a finite set V of elements called *vertices*, and a set E of pairs of vertices called *edges*. Let $V(G)$ represent the vertex set of G, and $E(G)$ represent the edge set of G. We call $adj(v)$ the *adjacency set* of vertex v, and we call the pair (v, w) an edge. Clearly $(v, w) \in E$ if and only if $w \in adj(v)$.

A subgraph H of G, denoted $H \subset G$, is a graph with $V(H) \subset V(G)$ and $E(H) \subset E(G)$. The degree of v in the subgraph H for any $v \in V(H)$, denoted $deg(v|H)$, is the number of vertices of H adjacent to v, i.e. $deg(v|H) = |adj(v|H)|$. An undirected graph G is called an *interval graph* if its vertices can be put into one-to-one correspondence with a set of intervals on the real line, such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection (have a common point). In other words, G is an interval graph provided that one can assign to each $v \in V(G)$ an interval I_v such that $I_u \cap I_v$ is nonempty if and only if $(u, v) \in E$.

An interval is defined by two points: left point (start of the interval) and right point (end of the interval). Let $l(I)$ and $r(I)$ correspond to the left and right

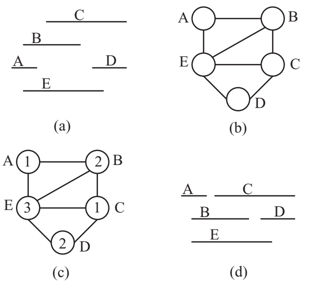


Fig. 4. Mapping batch scheduling to graph coloring

points of interval I respectively. Intervals I_v and I_u overlap if $l(I_v) \leq l(I_u) < r(I_v)$ or if $l(I_u) \leq l(I_v) < r(I_u)$.

A subgraph is called a *clique*, if every pair of vertices in the subgraph is connected by an edge. A clique C is maximal if there is no clique of G that properly contains C as a subset. Let $size(C)$ be the number of vertices in the clique, the maximum clique is the clique of greatest size among all maximal cliques.

One of the most important applications of interval graphs is job scheduling. Consider a set of n jobs to be scheduled on k servers. Finding a feasible schedule is equivalent to finding a proper k-coloring of the corresponding interval graph, such that no two adjacent vertices can have the same color (overlapping jobs are assigned to different servers). Interval graphs and graph coloring problems have been studied intensively in the literature (see [5, 6] and references within).

Batch scheduling in OBS networks can be treated as a subset of the interval graph coloring problem. We have a set of DBs that we want to assign to a number of wavelength channels. Each DB corresponds directly to an interval on the real line, and assigning wavelength channels to bursts is similar to assigning colors to intervals, where no two overlapping bursts can be assigned to the same wavelength channel. Figure 4 gives an example for mapping batch scheduling to graph coloring. In Figure 4a, a batch consisting of five bursts to be assigned to three wavelength channels. Figure 4b shows the corresponding interval graph. In Figure 4c, the interval graph is colored using three colors: 1, 2 and 3. Figure 4d shows the wavelength assignment of the batch.

Each CB represents a reservation request for an interval. The reservation request for DB b_i specifies a start time of the reservation $l(b_i)$, corresponding to the arrival time of the DB, and an end time of the reservation $r(b_i)$. The weight $w(b_i)$ of burst b_i is equal to $(r(b_i) - l(b_i))$.

4 Optimal Batch Scheduling Algorithm

The batch scheduling problem can be stated as follows: Given k wavelengths and a set of n bursts $\{b_1, .., b_n\}$ in the batch, find a feasible schedule that maximizes the value of the DBs accepted on the k wavelength channels. Note that n varies from one batch to another depending on the arrivals to the CB queue. The value of the accepted bursts is simply the sum of their weights. The weight of the burst is defined as the length of its corresponding interval.

A number of algorithms exist in the literature for finding an optimal feasible schedule of intervals while maximizing the total weight of the selected intervals [6, 7]. In [6], the authors formulated the problem as an instance of the interval graph coloring problem as explained next. Given k colors and n intervals, the objective is to maximize the value of the legally colored graph. The problem is then reformulated as the following binary integer linear program (ILP):

$$\begin{aligned} &\text{Maximize } w^T x \\ &\text{subject to } Ax \leq e^T k, \\ &\qquad\qquad x \in \{0,1\}, \end{aligned}$$

where w is an n-vector representing the weights of the intervals, x is a binary n-vector in which $x_j = 1$ implies that interval j is to be selected, $x_j = 0$ otherwise. A is $m \times n$ clique matrix [8], in which m is the number of the maximal cliques in the interval graph, where $A_{ij} = 1$ if interval j is in the i^{th} maximal clique, and $A_{ij} = 0$ otherwise. Also e is an m-vector of 1's. ILP problems are generally NP-hard. However, for our particular problem the A matrix has the *consecutive ones* property, and A is thus *totally unimodular* [9]. Therefore, we can relax the integrality constraint and solve the problem as a linear program in polynomial time.

This ILP formulation can be used to find an offline optimal schedule of bursts after receiving all the reservation requests. This optimal schedule can serve as a reference against which the performance of other algorithms can be compared.

Although it is tempting to use the same ILP formulation for finding the optimal schedule for a batch of bursts, it is not possible. The reason is that zero of more of the n bursts may not be assignable to one or more of the k wavelength channels due to already existing assignments of bursts belonging to previous batches. This constitute extra constraints in the problem.

To find an optimal batch schedule, we formulate our problem as an instance of the *scheduling with non-identical machines problem* [7], in which we associate with each interval (burst) a subset of servers (wavelength channels) on which it can be processed. Some bursts can not be processed on any wavelength channel, because all are busy in the time intervals corresponding to these bursts, these bursts will be dropped and not included in the optimization process. The authors in [7] have given an algorithm that finds the optimal feasible schedule for the non-identical machines problem. The algorithm works as follows. First build a list of events corresponding to the start and end times of the intervals being optimized. Then, for each event construct the vertices corresponding to legal combination of intervals and servers at the time of the event. The constructed vertices are used to build a directed graph. It then can be shown that the interval assignments along the longest path in the directed graph represent the optimal interval assignment to servers.

Unfortunately, the above algorithm has a $O(n^{k+1})$ computational complexity, which is high when k is large, rendering this algorithm impractical for use as an online batch scheduler. However, the algorithm is useful for comparison purposes since it serves as an upper bound on the performance of batch scheduling algorithms in terms of blocking probability.

5 Heuristic Batch Scheduling Algorithms

Because finding the optimal batch schedule has high computational complexity, we have to turn our attention to heuristic algorithms. In this section we propose a number of batch heuristic algorithms for wavelength channel scheduling in OBS networks. All of the proposed heuristic algorithms are based on performing the following two steps:

1. Impose a certain linear order on the control bursts in the batch queue.
2. Traverse the bursts in this linear order, and assign the corresponding data bursts to wavelength channels using a greedy void filling wavelength scheduling algorithm.

Although the ordering process imposes a small additional delay on the bursts in the batch queue, this delay is limited since the number of bursts in the batch queue is bounded by the acceptance delay value. If we were to order the bursts directly in the FIFO queue shown in Figure 2, then it is possible that a CB will be delayed past the arrival time of the corresponding DB because the number of bursts involved in the ordering process will not be bounded, and this shows the importance of the batch queue.

In the following we propose two batch ordering algorithms: *Smallest-Last Vertex Ordering,* and *Maximal Cliques First Ordering.*

5.1 Smallest-Last Vertex Ordering (SLV)

The SLV heuristic algorithm orders the vertices of the interval graph according to the smallest-last ordering [5]. The vertices $v_1, v_2, \ldots, v_n$ of a graph are said to be in smallest last order whenever v_i has minimum degree in the maximal subgraph on the vertices $v_1, v_2, \ldots, v_i$ for all i. Let $\delta(H) = min_{v \in V(H)}\{deg(v|H)\}$ be the minimum degree of graph H. A formal description of the SLV algorithm is given in Algorithm 1.

The SLV algorithm works as follows. Let v_n be chosen to have minimum degree in G. For $i = n-1, n-2, \ldots, 2, 1$, let v_i be chosen to have minimum degree in $H_i = \langle V(G) - v_n, v_{n-1}, \ldots, v_{i+1}\rangle$. From the resulting sequence $v_1, v_2, \ldots, v_n$, where v_n has the minimum degree in G, vertex v_1 will be colored first. Which means that the burst corresponding to v_1 will be assigned first to the first available wavelength. The process repeats until each burst is either assigned to a wavelength channel or dropped.

Choosing the largest degree vertices first for coloring would minimize the total number of colors required to produce proper coloring of the graph. This means

```
input      : $G$ on $n$ vertices
output     : An ordering $v_1, v_2, \ldots, v_n$ of vertices of G,
             where $deg(v_i|H_i) = \delta(H_i)$ for $1 \leq i \leq n$;
initialize $i \leftarrow n, H \leftarrow G$
while $i \geq 1$ do
    Let $v_i$ be a vertex of minimum degree in $H$;
    $H \leftarrow H - v_i$, $i \leftarrow i - 1$;
end
Report sequence $v_1, v_2, \ldots, v_n$;
```

Algorithm 1: SLV algorithm

that bursts which overlap the largest number of other bursts will be assigned first to wavelengths, which will lead to minimizing the number of bursts to be dropped. The algorithm does not consider the weight of the bursts, and it only attempts to minimize the *number* of dropped bursts.

5.2 Maximal Cliques First (MCF)

The basic idea behind the MCF algorithm is that since the maximum clique that can be colored is of size k, then our problem is equivalent to that of deleting a subset of intervals such that all remaining cliques are of size k or less. To this end, the MCF algorithm finds all the maximal cliques in the interval graph, then orders them in an increasing order according to time. Let $\{C_1, C_2 ... C_m\}$ be the set of maximal cliques in G ordered such that $C_i \prec C_j$ for $i < j$. The algorithm processes intervals belonging to C_i before intervals belonging to C_j for $i > j$. A formal description of the algorithm is given in Algorithm 2.

The MCF algorithm works as follows. It finds the list of maximal cliques in the interval graph with size larger than the number of available wavelengths. The algorithm then finds the clique with the latest occurrence time among this list. Thereafter, the algorithm removes the intervals with the smallest finish time from the maximal clique (and from the graph) such that the size of the maximal clique becomes equal to the number of the available wavelengths. This process is repeated until the size of all maximal cliques in the graph are less than or equal to the number of wavelengths. All vertices remaining in G are appended to the output list first, then the vertices removed by the algorithm are appended to the end of the output list.

6 Performance Evaluation

This section presents experimental results on the proposed algorithms: SLV and MCF, and their comparisons with the optimal batch algorithm (BATCH-OPT), and the greedy LAUC-VF algorithm. Our main concern in the following simulations is the blocking probability.

```
input      : $G$ on $n$ vertices, $k$
output     : A list of vertices $L$
initialize $H \leftarrow G$

while true do
    Let $C$ = list of all maximal cliques in $H$ with size $> k$;
    if $C$ is empty then
        break;
    end
    Let $c_{max}$ = clique with latest occurrence time in $C$;
    $z = size(c_{max}) - k$;
    for $i \leftarrow 0$ to $z - 1$ do
        $v_i$ = vertex with smallest finish time in $c_{max}$;
        $H \leftarrow H - v_i$;
        $c_{max} \leftarrow c_{max} - v_i$;
        $S \leftarrow S \cup v_i$;
    end
end
$L \leftarrow L \cup v_j, \forall v_j$ remaining in $G$;
Append $S$ to the end of $L$;
Report sequence $L$;
```

Algorithm 2: MCF algorithm

6.1 Simulation Setup

Figure 5 shows the simulation model used in this section, which is based on the OPNET [10] simulation tool. It consists of a single OBS node connected to traffic sources and a sink node. Each input link carries two separate wavelengths, one for data and one for control. The output link carries five wavelengths, four for data and one for control, and each wavelength has transmission speed of approximately 2.5 Gbps (OC-48). We assume that sources generate control and data bursts. Sources generate bursts according to Poisson process, and burst size has a mean value equal to B bits. The offset time has a uniform distribution over $[A_{min}, A_{max}]$. Let Δ be the transmission time of 1024 bits on one of the wavelengths, i.e. $\Delta = 1024/2377728000 = 4.3\text{e-}7$ seconds. We express the acceptance delay and the offset time in terms of Δ. In the following simulations, unless otherwise stated, we set the acceptance delay d to an arbitrary value of 100Δ second which is approximately 40μsec. The greedy wavelength algorithm used in conjunction with the batch algorithms is the LAUC-VF algorithm.

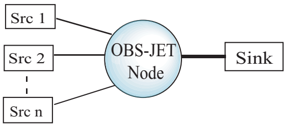


Fig. 5. Simulation model

6.2 Blocking Probability Versus Offered Load

In this section we study the effect of the proposed algorithms while varying the OBS node load. We study two scenarios. In the first scenario we use exponentially distributed burst sizes with mean value $B = 81920$ bits. The values of A_{max} and A_{min} are set to 150Δ second and 130Δ sec. respectively.

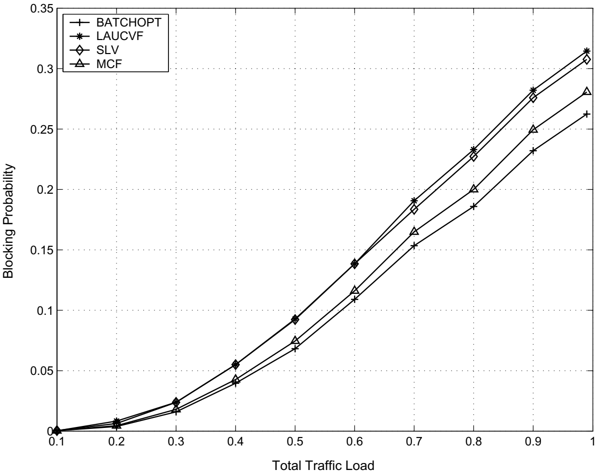


Fig. 6. Blocking Probability vs. Offered Load with exponential burst size

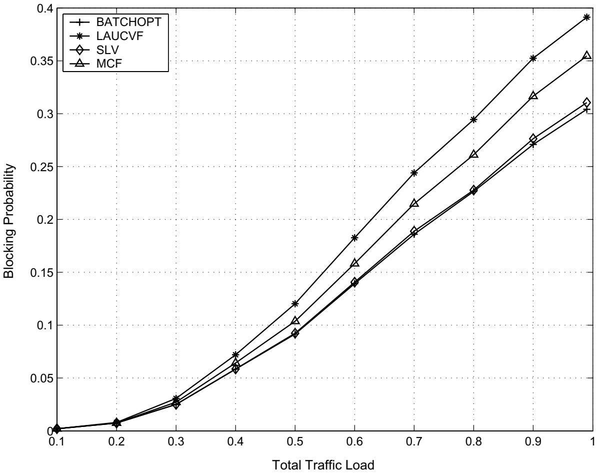


Fig. 7. Blocking Probability vs. Offered Load with constant burst size

Figure 6 shows that the performance of batch algorithms is upper bounded by the optimum batch algorithm as expected, and lower bounded by the greedy LAUC-VF algorithm. The figure shows that the MCF algorithm performs better than the SLV algorithm in this scenario, and its performance is close to the BATCH-OPT algorithm. In addition, the figure shows that the SLV algorithm performs slightly better than the LAUC-VF algorithm, however its results are not as significant when compared to the MCF algorithm.

In the second scenario we use constant burst size equal to 81920 bits. Figure 7 shows that the SLV algorithm is the best performing algorithms in this scenario. Moreover, the MCF algorithm in this scenario performs significantly better than the LAUC-VF algorithm, but not as good as the SLV algorithm.

6.3 Blocking Probability Versus Offset Time Range

We define the offset time range to be $A_{max} - A_{min}$. In this section, we vary the value of A_{min} to study effect of the offset time range on the performance of the batch scheduling algorithms. We use exponentially distributed burst sizes with mean $B = 81920$ bits. The value of A_{max} is set to 200Δ, and offered load is set to 99% of the link capacity. The value of A_{min} is varied between 10Δ and 150Δ. Figure 8 plots the blocking probability against the offset time range value.

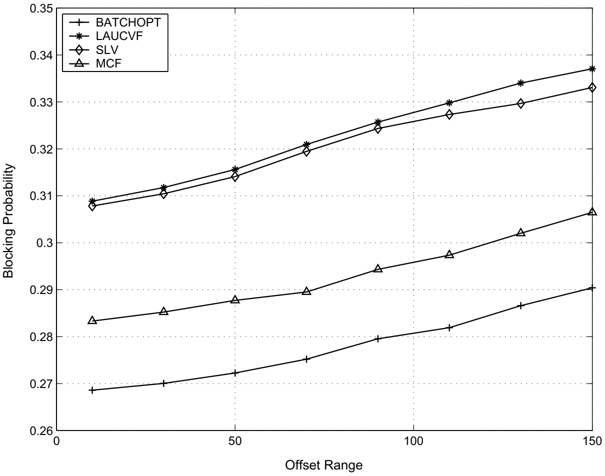


Fig. 8. Blocking Probability vs. Offset Time Range

Figure 8 illustrates that generally the blocking probability increases as the offset time range increases. Additionally, the rate of the increase in the blocking probability is approximately equal for all algorithms. This increase is due to the "retro-blocking" phenomena [11] of the JET signaling protocol, in which, a reservation request can be blocked by another reservation starting after its own time. Obviously this phenomena becomes more significant as the offset time range becomes larger.

7 Concluding Remarks

In this paper we have introduced a novel class of wavelength scheduling algorithms in OBS networks called batch scheduling algorithms. We have described an optimum batch scheduler that serves as an upper bound for the performance of the batch scheduling algorithms. Moreover, we have introduced two heuristic batch scheduling algorithms: SLV, MCF. The MCF algorithm was shown to be superior in scheduling variable size bursts. In case of constant size bursts, the SLV algorithm was shown to perform better than the MCF algorithm. Both algorithms reduce the blocking probability considerably with respect to greedy scheduling algorithms.

References

1. Qiao, C., Yoo, M.: Optical burst switching (obs) - a new paradigm for an optical internet. Journal of High Speed Networks **8** (1999) 69–84
2. Wei, J.Y., McFarland, R.I.: Just-in-time signaling for wdm optical burst switching networks. Journal of Lightwave Technology **18** (2000) 2019–2037
3. Kaheel, A., Alnuweiri, H.: Wavelength scheduling algorithms in buffer-less optical burst switching networks. In: Proceedings of IASTED International Multi-Conference on Wireless and Optical Communications-WOC'04, Canada (2004)
4. Xiong, Y., Vandenhoute, M., Cankaya, C.: Control architecture in optical burst-switched wdm networks. IEEE Journal on Selected Areas in Communications **18** (2000) 1838–1851
5. Matula, D.W.: Smallest-last ordering and clustering and graph coloring algorithms. Journal of the ACM **30** (1983) 217–427
6. Yannakakis, M.: The maximum k-colorable subgraph problem for chordal graphs. Information Processing Letters **24** (1987) 133–137
7. Arkin, M., Silverberg, E.: Scheduling jobs with fixed start and end times. Discrete Applied Mathematics **18** (1987) 1–8
8. Golumbic, M.C. In: Algorithmic Graph Theory and Perfect Graphs. Academic Press (1980)
9. Papadimitriou, C.H., Steiglitz, K. In: Combinatorial Optimization : Algorithms and Complexity. Dover Publications (1998)
10. OPNET Technologies Inc.: OPNET Modeler. Web page: `http://www.opnet.com/` (2005)
11. Kaheel, A., Alnuweiri, H., Gebali, F.: Analytical evaluation of blocking probability in optical burst switching networks. In: Proceedings of IEEE International Conference on Communications-ICC'04, France (2004)

ECC Based Threshold Cryptography for Secure Data Forwarding and Secure Key Exchange in MANET (I)

Levent Ertaul and Weimin Lu

Department of Math and Computer Science,
California State University, East Bay,
25800 Carlos Bee Blvd,
Hayward, CA 94542-3092 USA
lertaul@csuhayward.edu
wlu@horizon.csuhayward.edu

Abstract. This paper proposes a new approach to provide reliable data transmission in MANET with strong adversaries. We combine Elliptic Curve Cryptography and Threshold Cryptosystem to securely deliver messages in n shares. As long as the destination receives at least k shares, it can recover the original message. We explore seven ECC mechanisms, El-Gamal, Massey-Omura, Diffie-Hellman, Menezes-Vanstone, Koyama-Maurer-Okamoto-Vanstone, Ertaul, and Demytko. For secure data forwarding, we consider both splitting plaintext before encryption, and splitting ciphertext after encryption. Also we suggest to exchange keys between a pair of mobile nodes using Elliptic Curve Cryptography Diffie-Hellman. We did performance comparison of ECC and RSA to show ECC is more efficient than RSA.

1 Introduction

Mobile ad hoc networks are different from mobile wireless IP networks in that there are no base stations, wireless switches, and infrastructure services like naming, routing, certificate authorities, etc. Because mobile nodes join and leave the network dynamically, sometimes even without a notice, and move dynamically, network topology and administrative domain membership can change rapidly. Thus it is important to provide security services such as availability, confidentiality, authentication [1, 2], access control, integrity, and non-repudiation.

As in other networks, cryptography is the foundation for all network security services [3] in MANET, and key management is the major factor to guarantee a secure ad hoc network [4]. However, key management in ad hoc network has to be distributed service [5] as there is no fixed infrastructure to provide centralized service. Another major challenge is to deliver reliable data transmission when some nodes may be compromised [6]. Attackers can disrupt data transmission and incur significant data loss by tampering with, fraudulently redirecting, or even dropping data traffic.

First we suggest seven highly reliable data dispersing and data reconstruction mechanisms using ECC algorithms and TC. Then a key exchange mechanism using ECC Diffie-Hellman and TC is proposed as an alternative to RSADH.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 102 – 113, 2005.

2 Secure Data Transmission

We use Shamir's secret sharing scheme [7] to provide reliable data transferring. There are two basic mechanisms to combine ECC and TC. The first method is to split the messages into n pieces before we use ECC to encrypt them individually and send them to the receiver. At the receiving end, each share of secret is decrypted using ECC respectively. Then k piece shares in plaintext are combined to recover the secret. The second method is to encrypt the plaintext using ECC encryption algorithm before we split the ciphertext into n shares. The receiver with at least k shares of ciphertext is able to recover the ciphertext. Finally the destination can use ECC decryption algorithm to decrypt the ciphertext to get the original plaintext.

We can use any source routing protocols [8], like SDR, to find out the number of routes of disjoint nodes between the sender and the receiver to choose the number of shares n. Then, depending on n and the estimated number of compromised nodes in the network, we can come up with the number of share threshold k.

2.1 Transformation Between a Plaintext and a Point on Elliptic Curve

Koblitz [9] gives a method to convert a message to an elliptic curve point, and vice versa. We have a Galois Field GF(P), where P is a prime number and P > 3. An integer number k can used to determine how likely we are able to convert any plaintext into a corresponding point on elliptic curve. First we represent the message in ASCII code. Then we add a constant to each character in ASCII to get M such that kM < P. Next we search for an x such that there is a y and (x, y) is a point on the elliptic curve, where $kM < x \leq k(M + 1)$. To recover M, we compute $\lfloor (x - 1)/k \rfloor$. We notice that when k is larger, it is more likely to find a point (x, y) for the message M. But there always is a chance that there is not a corresponding point (x, y) for the message M such that $kM < x \leq k(M + 1)$.

2.2 ECC El-Gamal Cryptosystem

Elgamal works out an ingenious public key cryptosystem [11]. Suppose that the ECC has a point G on an elliptic curve $E_p(a, b)$, and the order of G is q. P is a large prime. Bob's private key and public key are n_B, $0 < n_B < q$, and $K_B = n_B G$.

2.2.1 Share Split Before Encryption

- First we choose a prime number p > max(M, n), and define $a_0 = M$, the message.
- Then we select k - 1 random, independent coefficients $a_1, a_2, \ldots, a_{k-1}$, $0 \leq a_j \leq p-1$, defining the random polynomial f(x) = $\sum_{j=0}^{k-1} a_j x^j$ over Z_p, a Galois prime field.
- We compute n shares, $M_t = f(x_t) \bmod p$, $1 \leq t \leq n$, where x_t can be just the public index t for simplicity, and convert them to points P_t on elliptic curve E_p (a, b).
- Alice picks a random number r, and sends rG and $P_t + rK_B$ to Bob with index t.
- Bob recovers each elliptic curve point by calculating $P_t + rK_B - n_B rG = P_t$.
- Bob converts P_t to M_t, and deduces M by using Lagrange interpolation formula

$$M = f(0) = \sum_{i=1}^{k} M_{t_i} \prod_{j=1}^{k} \frac{0- x_{t_j}}{x_{t_i} - x_{t_j}} = \sum_{i=1}^{k} M_{t_i} \prod_{j=1}^{k} \frac{-t_j}{t_i - t_j} \tag{1}$$

2.2.2 Share Split After Encryption

- Alice converts the secret M to a point P_M on the elliptic curve.
- Alice uses El-Gamal encryption to get $P_1 = rG$ and $P_2 = P_M + rK_B$.
- Let $P_2 = (x_2, y_2)$. We choose two random polynomials f_1, f_2 of degree k-1 in GF(p) such that $f_1(0) = x_2$, $f_2(0) = y_2$, and split x_2, y_2 into n shares of secret respectively. Alice sends P_1 and n shares of $P_2(x_2, y_2)$ with their corresponding indices to Bob.
- Bob recovers x_2, and y_2 , and calculates the point $P_M = P_2 – n_BP_1$.
- Eventually Bob will convert the point P_M to the secret M.

Instead of sending n pieces of x_2, y_2 to Bob, Alice can choose a random k – 1 degree polynomial f with $a_0 = x_2$ and $a_1 = y_2$. Thus Alice and Bob can use Vandermonde matrix [10] instead of Lagrange interpolation to share more than one secretes.

2.3 The Massey-Omura Protocol

Since Massey-Omura encryption [12] requires four transmissions between Alice and Bob, it is not an efficient solution for threshold crypto-system. Let N be the order of $E_p(a, b)$. Alice and Bob choose their secret key n_A and n_B respectively, such that $\gcd(n_A, N) =1$ and $\gcd(n_B, N) =1$. $NR = \propto$ for any point R on the curve according to Lagrange's theorem. $n_B^{-1}P_{M_3} = n_B^{-1}n_Bn_A^{-1}n_AP_M = n_B^{-1}n_B(P_M + qNP_M) = n_B^{-1}n_B P_M = P_M$

Encryption algorithm:

- Alice calculates plaintext M's corresponding point P_M on the elliptic curve, and sends $P_{M_1} = n_AP_M$ to Bob.
- Bob sends back $P_{M_2} = n_BP_{M_1}$ to Alice.
- Alice sends $P_{M_3} = (n_A^{-1} \bmod N)P_{M_2}$ to Bob.

Decryption algorithm:

Bob calculates $(n_B^{-1} \bmod N)\, P_{M_3} = P_M$, and recovers plaintext M by P_M.

2.3.1 Share Split Before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice converts a share M_t into a point P_t on the curve, and sends $P_{t_1} = n_AP_t$ to Bob.
- Bob sends $P_{t_2} = n_BP_{t_1}$ to Alice.
- Alice computes $P_{t_3} = n_A^{-1}P_{t_2}$ and sends it to Bob, $n_A^{-1} \in Z_N$.
- Bob computes $n_B^{-1}P_{t_3}$ and it is P_t, $n_B^{-1} \in Z_N$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.3.2 Share Split After Encryption

- Alice converts the secret M to a point $P_M = (x, y)$ on the curve.
- Alice computes $P_{M_1} = n_AP_M = (x_1, y_1)$, splits x_1 and y_1 into n shares respectively, and sends n pieces of x_{1_t} and y_{1_t} to Bob.

- Bob combines k pieces of x_{1_t} and y_{1_t} separately to get (x_1, y_1), i.e., P_{M_1}, computes $P_{M_2} = n_A P_{M_1} = (x_2, y_2)$, splits x_2 and y_2 into n shares respectively, and sends n pieces of x_{2_t} and y_{2_t} to Alice.
- Alice combines k pieces of x_{2_t} and y_{2_t} separately to get (x_2, y_2), i.e., P_{M_2}, computes $P_{M_3} = n_A^{-1} P_{M_2} = (x_3, y_3)$, splits x_3 and y_3 into n shares respectively, and sends n pieces of x_{3_t} and y_{3_t} to Bob.
- Bob combines k pieces of x_{3_t} and y_{3_t} separately to get (x_3, y_3), i.e., P_{M_3}, computes $P_M = n_B^{-1} P_{M_3}$, and converts the point P_M to the secret M.

2.4 ECC Diffie-Hellman Protocol

A generalization of the original Diffie-Hellman key exchange in Z_p^* found a new depth when Koblitz [13] suggested that such a protocol could be used with the group over an elliptic curve. The order of a point G on an elliptic curve $E_p(a, b)$ is q. P is a large prime. The secret key $K = n_A n_B G$ is generated using DH algorithm.

Encryption algorithm:

- Alice finds the point P_M corresponding to M, and sends $P_M + n_A n_B G$ to Bob.

Decryption algorithm:

- Bob subtracts $n_A n_B G$ from $P_M + n_A n_B G$, and converts P_M to the plaintext M.

2.4.1 Share Split Before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \leq t \leq n$.
- Alice converts one share M_t to a point P_t on the curve.
- Alice computes $P_t + n_A n_B G$ and sends it to Bob.
- Bob recovers P_t by subtracting $n_A n_B G$ from $P_t + n_A n_B G$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.4.2 Share Split After Encryption

- Alice converts the secret M to a point $P_M = (x, y)$ on the curve.
- Alice computes $P_C = n_A n_B G + P_M = (x_C, y_C)$.
- Alice splits x_C and y_C into n shares of x_{C_t} and y_{C_t} respectively, $1 \leq t \leq n$.
- Alice sends n pieces of x_{C_t} and y_{C_t} to Bob.
- Bob combines k pieces of x_{C_t} and y_{C_t} separately to get (x_C, y_C), i.e., P_C.
- Bob computes $P_M = P_C - n_A n_B G$, and converts the point P_M to the secret M.

2.5 The Menezes-Vanstone Cryptosystem

Menezes-Vanstone Elliptic Curve Cryptosystem [14] is a solution to the problem of encoding a message in a point. It uses a point on an elliptic curve to mask a point in the plane. It is fast and simple. Let H be a cyclic subgroup of $E_p(a, b)$ with the generator G. Bob has a private key n_B, and a public key $n_B G$. The message *M* is converted into a point $P_M = (x, y)$ in *GF(p)*.

Encryption algorithm:

- Alice select a random number r < |H|, and calculates $rn_BG = (x_k, y_k)$.
- Alice sends $(rG, x_kx \ mod \ p, y_ky \ mod \ p)$ to Bob.

Decryption algorithm:

- Bob calculates $n_BrG = rn_BG = (x_k, y_k)$.
- Bob recovers x and y by $x_k^{-1}x_kx \ mod \ p$ and $y_k^{-1}y_ky \ mod \ p$.
- Bob converts the point(x, y) to get the original plaintext M.

2.5.1 Share Split Before Encryption

- Alice splits the message M into n shares of secret M_t, $1 \le t \le n$.
- Alice converts each share M_t into a point P_t.
- Alice select a random number r < |H|, and calculates $rn_BG = (x_k, y_k)$.
- Alice sends $(rG, x_kx_t \ mod \ p, y_ky_t \ mod \ p)$ to Bob.
- Bob calculates $n_BrG = rn_BG = (x_k, y_k)$.
- Bob recovers x_t and y_t by $x_k^{-1}x_kx_t \ mod \ p$ and $y_k^{-1}y_ky_t \ mod \ p$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.5.2 Share Split After Encryption

- Alice converts the message M into a point P_M.
- Alice select a random number r < |H|.
- Alice calculates $rn_BG = (x_k, y_k)$, and calculates $z = x_kx \ mod \ p$, and $w = y_kx \ mod \ p$.
- Alice splits z, w into n shares of z_t, and w_t respectively, $1 \le t \le n$.
- Alice sends rG and n pieces of z_t, and w_t to Bob.
- Bob combines k pieces of z_t and w_t separately to get (z, w).
- Bob calculates $n_BrG = rn_BG = (x_k, y_k)$.
- Bob recovers P_M by $x_k^{-1}z = x_k^{-1}x_kxmod \ p$ and $y_k^{-1}w = y_k^{-1}y_ky \ mod \ p$.
- Eventually Bob converts P_M to the secret M.

2.6 The Koyama-Maurer-Okamoto-Vanstone Cryptosystem

KMOV [15] conjugates the polynomial-time extraction of roots of polynomials over a finite field with the intractability of factoring large numbers. Bob chooses two large prime numbers, p and q, such that $p \equiv q \equiv 2$ mod 3. Let n = pq, 0 < b < p and b < q, and N = lcm(p+1, q+1). Bob picks up his public key e with gcd(e, N) = 1. His private key d is e^{-1} mod N.

Encryption algorithm:

- Alice represents M as a point P_M on elliptic curve $E_n(0, b)$, and sends eP_M to Bob.

Decryption algorithm:

- Bob calculates $deP_M = (rN + 1) P_M = P_M$, where r is an integer.
- Bob recovers the original plaintext M by P_M.

2.6.1 Share Split Before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice converts a piece of share M_t into a point P_t on the curve.
- Alice computes eP_t and sends it to Bob.
- Bob recovers P_t by $deP_t = P_t$.
- With at least k share of P_M, Bob recovers P_M, and converts the P_M to the secret M.

2.6.2 Share Split After Encryption

- Alice converts the secret M to a point $P_M = (x, y)$ on the curve.
- Alice computes $P_C = eP_M = (x_C, y_C)$.
- Alice splits x_C and y_C into n shares of x_{C_t} and y_{C_t} respectively, $1 \le t \le n$.
- Alice sends n pieces of x_{C_t} and y_{C_t} to Bob.
- Bob combines k pieces of x_{C_t} and y_{C_t} separately to get (x_C, y_C), i.e., P_C.
- Bob computes $P_M = dP_C$, and converts the point P_M to the secret M.

2.7 The Ertaul Crypto-system

P is the generator point while x is the private key, and Y = x*P is the public key. $H((xi, yi)) = Hash(xi \oplus yi)$ is a HASH function such as MD5, SHA-1.

Encryption algorithm:

1. Alice selects a random value r from Z_q.
2. Alice computes U = r*P and $V = H(r*Y) \oplus M$, and sends C = (U, V) to Bob.

Decryption algorithm:

1. Given a ciphertext C = (U, V), Bob computes x*U= x*r*P = r*x*P.
2. Bob computes $V \oplus H(r*x*P) = H(r*Y) \oplus M \oplus H(r*x*P) = M$.

2.7.1 Share Split Before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- Alice selects a random value r from Z_q, and computes U = r*P.
- For each share M_t, Alice computes $V_t = H(r*Y) \oplus M_t$.
- Alice sends ciphertext C_t = (U, Vt) to Bob.
- Given a ciphertext C_t, Bob computes x*U= x*r*P.
- Bob computes H(r*x*P) and $Vt \oplus H(r*x*P) = H(r*Y) \oplus M_t \oplus H(r*x*P) = M_t$.
- With at least k share of M_t, Bob is able to recover M.

2.7.2 Share Split After Encryption

1. Alice selects a random value r from Z_q, computes U = r*P.
2. Alice computes $V = H(r*Y) \oplus M$, splits V into n shares of secret V_t, $1 \le t \le n$.
3. Alice sends ciphertext $C_t = (U, V_t)$ to Bob.
4. Bob recovers V, and computes x*U= x*r*P.
5. Bob computes H(x*r*P) and $V \oplus H(x*r*P) = H(r*Y) \oplus M \oplus H(x*r*P) = M$.

2.8 The Demytko Cryptosystem

Demytko [16] uses a fixed randomly chosen elliptic curve $E_n(a, b)$ over the ring $\mathbb{Z}_n$, where n = pq is an RSA modulus. It relies on the fact that if a number x is not the x-coordinate of a point on an elliptic curve $E_p(a, b)$, then it will be the x-coordinate of a point of the twisted curve $\overline{E_p(a,b)}$ defined as, in addition to the point at infinity, the set of points (x, y) satisfying $\overline{E_p(a,b)}$: $y^2 = x^3 + ax + b$, where $y = u\sqrt{v}$, $u \in F_p$, and v is a fixed quadratic non-residue modulo p. Let $|E_p(a, b)| = 1 + p + \alpha$, $|\overline{Ep(a, b)}| = 1 + p - \alpha$, $|E_q(a, b)| = 1 + q + \beta$, $|\overline{Eq(a, b)}| = 1 + q - \beta$.

$N_1 = \mathrm{lcm}(p + 1 + \alpha , q + 1 + \beta)$ $N_2 = \mathrm{lcm}(p + 1 + \alpha , q + 1 - \beta)$

$N_3 = \mathrm{lcm}(p + 1 - \alpha , q + 1 + \beta)$ $N_4 = \mathrm{lcm}(p + 1 - \alpha , q + 1 - \beta)$

e is chosen such that $\gcd(e, N_i) = 1$, and private key d_i is calculated by $d_i = e^{-1} \bmod N_i$, i = 1 to 4. Let x represent the plaintext and s the ciphertext (where $0 \le x, s \le n - 1$).

Encryption algorithm:

Alice sends (s, t) = e(x, y) where $y = \sqrt{x^3 + ax + b}$ and $t = \sqrt{s^3 + as + b}$.

Decryption algorithm:

Bob determines which d_i of the four inverses of e should be used based on the Jacobi symbols $(c^3 + ac + b/p)$ and $(c^3 + ac + b/q)$. Bob computes(x, y) = d_i(s, t).

2.8.1 Share Split Before Encryption

- Alice splits the secret M into n shares of secret M_t, $1 \le t \le n$.
- For each share M_t, Alice computes $C_t = e(x_t, y_t)$, and sends ciphertext C_t to Bob.
- Given a ciphertext C_t, Bob chooses which d of the four inverses of e to be used.
- Bob computes $(x_t, y_t) = dC_t$, and recovers M.

2.8.2 Share Split After Encryption

- For a message M, Alice computes C = e(M, y).
- Alice splits C into n shares of secret C_t, $1 \le t \le n$, and sends ciphertext C_t to Bob.
- With at least k share of C_t, Bob is able to recover C.
- Bob chooses which d of the four inverses of e to use, and computes (M, y) = dC.

3 Key Exchange Method

Suppose that the order of a point G on an elliptic curve $E_p(a, b)$ is q. P is a large prime. Reliable key exchange using Threshold Crypto-systems works like below.

- First, Alice chooses a secret number n_A with $0 < n_A < q$.
- Bob chooses a secret number n_B with $0 < n_B < q$.
- Alice computes her public key $K_A = (x_A, y_A) = n_AG$.
- Alice splits x_A and y_A into n separate shares, x_{A_t} and y_{A_t}, and sends them to Bob.
- Bob computes his public key $K_B = (x_B, y_B) = n_BG$.

- B splits x_B and y_B into n separate shares, x_{B_t} and y_{B_t}, and sends them to Alice.
- Alice uses k shares of x_{B_t} and y_{B_t} separately to recover x_B, and y_B, i.e., n_BG.
- Alice calculates secret key $K = n_An_BG$.
- Bob uses k shares of x_{A_t} and y_{A_t} separately to recover x_A, and y_A, i.e., n_AG.
- Bob computes the same key by $n_Bn_AG = n_An_BG$.

4 Performance and Complexity Comparison

4.1 Computation Complexity of Share Split Before Encryption and Share Split After Encryption

There are mainly three types of operations in our methods, point addition, point exponentiation and Lagrange interpolation. Doubling of points takes one inverse operation, 10 additions, and six multiplications while addition of two different points takes one inverse operation, 10 additions, and five multiplications. Addition points and doubling of points of basic ECC arithmetic are of comparable computation complexity.

Next we will compare the complexity of point addition and point exponentiation, i.e., P + Q and rG. Let w = p's length in bits and r is a number in GF(p). We can represent r in binary as $r_wr_{w-1}\ldots r0$ or $r = \sum_{i=1}^{w} 2^{i-1} r_i$. $rG = (\sum_{i=1}^{w} 2^{i-1} r_i)G$

Since integer multiplications are much more expensive than integer additions, from now on, we will take only multiplications into consideration. w - 1 additions of two different points need 5(w – 1) multiplications and w – 1 inverse operations. We can see that addition of two points is far cheaper than exponentiation of a point. A (k, n) Shamir's secret sharing algorithm has a complexity of $O(k^2)$, k^2 multiplications and $k^2 + k$ additions. As k << w ($160 \le w \le 256$), it is straightforward to see that ECC encryption/decryption is much more expensive than Lagrange interpolation in the same prime field. Table 1 lists the number of three types of calculations required for each ECC-TC algorithm plus number of packets and packet sizes between senders and receivers.

w is the length in bits needed to represent the largest number that can be used, i.e., p, and w = $\lceil \log p \rceil$. Each packet contains all necessary information for each round of communication between the sender and the receiver, including each individual share x_i and its index i. >From this table, we can see, in general, share split before encryption is slower than share split after encryption. Among seven ECC-TC algorithms, MV and Ertaul are the most best from the perspective of computing power requirements because they have the least number of elliptic curve exponentiation calculations over prime fields. If network bandwidth is the critical factor, KMOV and Demytko are better choices.

Table 1. Complexity comparison of seven ECC secret sharing algorithms

ECC	Share split before encryption			Share split after encryption			Pkt size	Pkt #
	rG	P+Q	Lagrange	rG	P+Q	Lagrange		
EG	3n	2n	1	3	2	2	5w	n
MO	4n	0	1	4	0	6	3w	3n
DH	0	2n	1	0	2	2	3w	n
MV	3	0	1	3	0	2	5w	n
KMOV	2n	0	1	2	0	2	3w	n
Ertaul	3	0	1	3	0	1	4w	n
Demytko	2	0	1	2	0	1	3w	n

4.2 Performance Advantages Comparison of ECC with RSA

Currently, for the same level of resistance against the best known attacks, the system parameters for an elliptic-curve-based system can be chosen to be much smaller than the parameters for RSA or mod p systems [17]. Table 2 and 3 are taken from [17, 18], and are directly comparable to RSA numbers for the same platform. Table 3 uses ECCDH for ECC encryption and decryption. It calculates the time taken to compute the secret key n_An_BG. Encryption and decryption time are symmetric in ECCDH. Table 4 is from [19]. In table 5, we calculate the timings of seven ECC secret sharing algorithms by only considering point exponentiation since that is bulk of the calculation. Clearly ECCDH is the fastest algorithm. Same as before splitting secret after encryption has better performance than splitting secret before encryption.

Table 2. Key sizes in bits for equivalent levels

Symmetric	ECC	DH/DSA/RSA
80	163	1024
128	283	3072
192	409	7680
256	571	15360

Table 3. Sample ECC and RSA timings in milliseconds over prime fields

Processor	MHz	163-ECC	192-ECC	1024-RSAe	1024-RSAe	2048-RSAe	2048-RSAd
UltraSPARC II	450	6.1	8.7	1.7	32.1	6.1	205.5
StrongARM	200	22.9	37.7	10.8	188.7	39.1	1273.8

Table 4. ECC of Koblitz curve over F_{2^m} and RSA ($e = 2^{16} + 1$) timings in milliseconds on Pentium II 400MHZ

163-ECAESe	163-ECAESd	233-ECAESe	233-ECAESd	283-ECAESe	283-ECAESd
4.37	2.85	7.83	4.85	11.02	6.78
1024-RSAe	1024-RSAd	2048-RSAe	2048-RSAd		
3.86	66.56	13.45	440.69		

Table 1 and Table 5 clearly indicate share split before encryption is not as efficient as share split after encryption. The reason is that share split before encryption splits a secret into n shares and does n encryptions instead of just one in share split after encryption. If the environment has limited computing power, like mobile network or embedded system, MV, KMOV and Demytko take 2 elliptic curve exponentiations while other algorithms requires at least 3. On the other hand, if the application is running over a network with very limited capacity, packet size is more important than complexity. In that case, Massey-Omura, KMOV, and Demytko are the best choices as the packet size is 3w, 3w, and 2w respectively. We noticed that Ertaul ECC-TC has the following advantages: Converting messages into points eliminated, ECC become a block cipher, proven secure cryptosystems (ECC and MD5/SHA-1). Furthermore it is a stable algorithm, i.e., it has the same complexity in both share split before encryption and share split after encryption. It is close to the best candidates in all cases.

Table 5. ECC secret sharing timings in milliseconds over prime fields

ECC	share split before encryption				share split after encryption			
	163-bit Sun	192-bit Sun	163-bit ARM	192-bit ARM	163-bit Sun	192-bit Sun	163-bit ARM	192-bit ARM
EG	18.3n	26.1n	68.7n	113.1n	18.3	26.1	68.7	113.1
MO	24.4n	34.8n	91.6n	150.8n	24.4	34.8	91.6	150.8
DH	6.1	8.7	22.9	37.7	6.1	8.7	22.9	37.7
MV	12.2	17.4	45.8	75.4	12.2	17.4	45.8	75.4
KMOV	12.2n	17.4n	45.8n	75.4n	12.2	17.4	45.8	75.4
Ertaul	18.3	26.1	68.7	113.1	18.3	26.1	68.7	113.1
Demytko	18.3n	26.1n	68.7n	113.1n	12.2	17.4	45.8	75.4

5 Related Works

A different approach has been proposed to alleviate the detrimental effects of packet dropping by detecting misbehaving nodes and reporting such events, and maintaining a set of metrics reflecting the past behavior of other nodes [20]. It consists of two entities, the watchdog and the pathrater. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. If the next node does not forward the packet, then it is malicious. The pathrater uses this feedback to choose the best route that is most likely to deliver packets.

Secure Message Transmission Protocol [21] is a comprehensive protocol that tolerates rather than detects and isolates malicious nodes. SMT requires a security association between the two end communicating nodes, i.e., the source and the destination. It uses Active Path Set, a set of diverse, node disjoint paths to transfer dispersed pieces of each outgoing message using Information Dispersal Algorithm. SMT can operate with any underlying secure routing protocol. The message and redundancy data are divided into a number of pieces so that if M out of N transmitted pieces are received successfully, the original message can be correctly reconstructed. The sender updates the rating of each path in its APS based on the feedback provided by the destination. The destination validates the incoming pieces and acknowledges the successfully received ones through a feedback across multiple routes back to the source.

We need to implement our solution and run it by a simulator to compare it with other related works to see the performance improvement.

6 Conclusions

The security-sensitive applications of ad hoc networks require high degree of security, but ad hoc networks are inherently vulnerable to security attacks. Threshold cryptography is a valid approach to build a highly available and highly secure key management service by distributing trust among a group of servers. Elliptic curve cryptography provides an efficient alternative to RSA public key encryption. We successfully use ECC and TC to provide the best of both worlds in MANET environment.

References

[1] T.P. Pedersen, "A Threshold Cryptosystem without a Trusted Party", In *Proc. Of Eurocrypt'91*, Lecture Notes in Computer Science, LNCS 547, Springer Verlag, pp.522-526, 1991

[2] Kazuo Takaragi, Kunihiko Miyazaki, Masashi Takahashi, *A Threshold Digital Signature Issuing Scheme without Secret Communication*

[3] Amitabh Mishra and Ketan Nadkarni, Security in Wireless Ad Hoc Networks, *The Handbook of Ad Hoc Wireless Networks,* December 2002, pp. 30.1-30.51

[4] L. Zhou and Z.J. Haas, Securing Ad Hoc Networks, *IEEE Network Magazine*, Nov./Dev. 1999

[5] Dan Zhou, Security Issues in Ad Hoc Networks, The *Handbook of Ad Hoc Wireless Networks,* December 2002, pp. 32.1-30.14

[6] Panagiotis Papadimitratos and Zygmunt Haas, Securing Mobile Ad Hoc Networks, *The Handbook of Ad Hoc Wireless Networks,* December 2002, pp. 31.1-31.17

[7] A. Shamir, How to Share a Secret, in *Communications of the ACM, vol.22, no.11*, pp.612-613, 1979

[8] P. Papadimitratos, Z.J. Haas, Secure Routing for Mobile Ad Hoc Networks, in: *Proceddings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 27-31, 2002

[9] N. Koblitz, *A Course in Number Theory and Cryptography (Graduate Texts in Mathematics, No 114)*, Springer-Verlag, 1994

[10] W. Trappe, L. C. Washington, *Introduction to Cryptography: with Coding Theory*, Prentice Hall, 2002

[11] T. Elgamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transactions on Information Theory,* IT-31(4):469-472, July 1985

[12] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, 2003

[13] N. Koblitz, Elliptic Curve Cryptosystems, *Math. Comp.*, 48,203-209, 1987

[14] A. Menezes, S. A. Vanstone, Elliptic Curve Cryptosystems and Their Implementation, *Journal of Cryptology*, 6 (1993), 209-224

[15] K. Koyama, U. Maurer, T. Okamoto, S. A. Vanstone, New Public-Key Schemes Based on Elliptic Curves over the Ring Z_n, *Proceedings of Crypto'91*, LNCS 576, Springer-Verlag, pp. 252-266, 1992

[16] N. Demytko, A New Elliptic Curve Based Analogue of RSA, *EUROCRYPT'93*, LNCS 765 40-49 (1993)

[17] Kristin Lauter, The Advantages of Elliptic Curve Cryptography for Wireless Security, IEEE Wireless Communications, February 2004

[18] V. Gupta, S. Gupta, S. Chang, Performance Analysis of Elliptic Curve Cryptography for SSL, ACM Workshop Wireless Security, Mobicom 2002, Atlanta, GA, September 2002

[19] Michael Brown, Donny Cheung, Darrel Hankerson, Julio Lopez Hernandez, Michael Kirkup, and Alfred Menezes, PGP in Constrained Wireless Devices, in: *Proceedings of the 9th USENIX Security Symposium*, Denver, CO, August 2000

[20] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigaing Routing Misbehavior in Mobile Ad Hoc Networks, in: *Proceedings of the 6th MobiCom*, Boston, MA, August 2000

[21] P. Papadimitratos, Z.J. Haas, Secure Message Transmission in *Mobile Ad Hoc Networks*

Mutual Authentication and Key Exchange Protocols with Anonymity Property for Roaming Services

Yixin Jiang[1], Chuang Lin[1], Xuemin Shen[2], and Minghui Shi[2]

[1] Department of Computer Science and Technology, Tsinghua University,
Beijing, 100084, P. R. China
Tel: +86(10)6279-6495, 6278-3596, Fax: +86(10)6277-1138
{yxjiang, clin}@csnet1.cs.tsinghua.edu.cn
http://qos.cs.tsinghua.edu.cn

[2] Department of Electrical and Computer Engineering, University of Waterloo,
Waterloo, Ontario, N2L 3G1, Canada
Tel: +1(519)888-4567 x 2691, 7473, Fax: +1(519)746-3077
{xshen, mshi}@bbcr.uwaterloo.ca
http://bbcr.uwaterloo.ca/~xshen

Abstract: Two novel mutual authentication and key exchange protocols with anonymity are proposed for different roaming scenarios in the global mobility network (GLOMONET). The proposed protocols have new features, such as identity anonymity and one-time session key progression. Identity anonymity protects mobile users' privacy in the roaming network environment. One-time session key progression frequently renews the session key for mobile users and reduces the risk of using a compromised session key to communicate with visited networks. It is shown that the computation complexity of the proposed protocols is similar to the existing one appeared in the literature, while the security has been significantly enhanced.

Index Terms: authentication, key exchange, roaming service, anonymity, secret-splitting, self-certified.

1 Introduction

Global mobility network (GLOMONET) [1], such as GSM and CDMA etc., increases the possibility of illegal access from a malicious intruder while offering more effective global roaming service for a legitimate user between the home network and the visited network. Several authentication protocols for global roaming service have been developed in the GLOMONET [2]. Suzuki et al developed an authentication protocol for roaming service [1]. They introduced a challenge/response interactive authentication mechanism with a symmetric cryptosystem to construct their authentication protocol. Buttyan et al pointed out some potential attacks to the authentication protocol in [1], and further proposed an improved protocol and made it resistant against the presented attacks [3]. Subsequently, Hwang et al [4] introduced a new self-encryption mechanism to simplify the protocol in [3].

However, in [4], the long-term key K_{MH} shared between home network H and user M is calculated as $K_{MH} = f(ID_M)$, where f is assumed to be a secret one-way function.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 114 – 125, 2005.

The protocol cannot provide identity anonymity, and an intruder can easily obtain ID_M by intercepting the messages. If the function f is spied, the intruder can compute corresponding K_{MH} of each user, which comprises the whole cryptographical infrastructure and then the advantage of self-encryption would be counteracted. The disclosure of a user identity will allow unauthorized entities to track his moving history and current location. Any illegal access to information related to the user location without his notice can be a serious violation of his privacy. Hence, the identity anonymity is one important property that should be considered for roaming services. The proposed authentication protocols use the temporary identity (TID) for a mobile user instead of his real one. TID is prearranged and distributed by the home network H in advance or generated by encrypting the real identity.

A secure protocol design for roaming services requires, (1) Mutual authentication between a network entity and a mobile user; (2) Mutual agreement of shared session key; (3) Assuring the freshness of session key; (4) Mutual implicit key authentication [5]. Since the protocols are implemented on the mobile device used in wireless environment, there are other two factors to be considered. Firstly, the low computational power of mobile devices should be a concern, which means a security protocol requiring heavy computation on the mobile is not feasible [6, 7]. Secondly, since the bandwidth is lower and the channel error is higher in wireless networks than that in wired networks, the security protocols should be designed to minimize the message size and the number of message exchanges.

In this paper, aiming at providing the identity anonymity and simplifying the existing authentication protocol for secure roaming service in GLOMONET environment, we propose two sets of mutual authentication and key exchange protocols with anonymity property for roaming service. The first proposed protocol uses the secret-splitting principle. The other uses self-certified scheme [8, 9], known as a public key authentication cryptosystem. The two protocols can be deployed depending on whether the home network has setup a long-term secret key with its users. The mutual authentication with anonymity property prevents the disclosure of mobile users' real identities and protects their privacy in the roaming network environment. The key exchange renews a mobile user's session key for each session, and therefore, reduces the risk of using a compromised session key to communicate with visited networks. Although with enhanced security features, the proposed protocols require similar computation power as the existing protocols.

The rest of this paper is organized as follows. In Sections 2and 3, two new authentication and key exchange protocols with anonymity for secure roaming service are proposed respectively. In Section 4, the performance comparisons between the protocol in [4] and the proposed two protocols are presented in detail, followed by conclusions in Section 5.

2 Protocol I for Secure Roaming Services

The proposed protocol I for secure roaming service is based on the *secret splitting* principle [10]. The protocol includes two phases. In phase I, the visited network V authenticates a roaming user M through his home network H. After certification, an authentication key is established between *M* and *V*. In the subsequent communications, *V* can directly authenticate *M* by using the authentication key rather than authenticating it through *H*. In phase II, user *M* establishes a session key with *V*. Then, *M* can directly visit *V* and *V* can provide services for *M*. A novel mechanism, called "one-time session key progression", assures the mutual authentication and the freshness of session key. The proposed protocol uses symmetric encrypt algorithm and can be applied when visited network and home network have pre-setup shared secret.

2.1 Phase I: Mutual Authentication with Anonymity Property

Let *H* generate an *m*-bits random number *N* and keep it secretly. Note that in order to prevent the exclusive search attack, *m* should be sufficiently large, e.g., 256 bits. When user *M* registers with his home network *H*, he submits his identity ID_M to *H*. Then, *H* computes a pseudonym identity PID_M for user *M* as follows:

$$PID_M = h(N \parallel ID_H) \oplus ID_M \oplus ID_H \tag{1}$$

where "$\oplus$" denotes bitwise XOR operation, and *h* is a public strong one-way hash function. Subsequently, *H* delivers PID_M to *M* through a secure channel, e.g., *H* can issue a smart card for user *M*. By this simple secret-splitting mechanism, the real identity ID_M can be concealed in PID_M and identity anonymity for *M* can be provided without increasing the algorithm complexity.

The detailed steps of the proposed mutual authentication protocol for the roaming services (Phase I) is described in Fig. 1. A simple secret splitting mechanism is introduced to provide the identity anonymity and prevent unauthorized entities from tracing the mobile user's roaming history and his current location (Step 3). The authentication key is computed with the random numbers chosen by *M* and *V* by

$$K_{auth} = r_M \oplus r_V. \tag{2}$$

The mechanism makes the protocol fairer and more secure without increasing the computation complexity because the XOR is a very low time-consuming operation. In addition, the self-encryption property of the protocol in [4] is still remained. The home network maintains a long-term secret key $K_{MH} = f(ID_M)$ for his user *M* by using a one-way function. By extracting the real identity ID_M of user *M* from the pseudonym identity PID_M, we can generate the shared key K_{MH}, which is used to encrypt the corresponding cipher-text.

Message 1. $M \rightarrow V$: $ID_H, PID_M, E_{K_{MH}}(r_M \parallel K_{MH})$

Message 2. $V \rightarrow H$: $PID_M, E_{K_{VH}}(r_V \parallel T_V \parallel E_{K_{MH}}(r_M \parallel K_{MH}))$

Message 3. $V \leftarrow H$: $E_{K_{VH}}(r_V \parallel r_M \parallel h(ID_M)),\ E_{K_{MH}}(r_M \parallel r_V \parallel ID_V)$

Message 4. $M \leftarrow V$: $E_{K_{MH}}(r_M \parallel r_V \parallel ID_V)$

Message 5. $M \rightarrow V$: $E_{K_{auth}}(K_{auth})$

Fig. 1. Authentication Protocol Based on Secret Splitting Principle

In the following, we describe the proposed authentication protocol following the order of message exchange, and discuss the security goals which can be achieved during the execution of each protocol message.

Step 1) When a mobile user M enters a new visited network V, M initiates a registration authentication process with V to identify himself to be a legal subscriber of his home network H. M does the following: (1) Generate a secret random r_M ; (2) Compute his long-term secret key $K_{MH} = f(ID_M)$ and $E_{K_{MH}}(r_M \| K_{MH})$; (3) Send $E_{K_{MH}}(r_M \| K_{MH})$, PID_M, and ID_H to V.

Step 2) On receiving message 1 from M, V forwards PID_M and sends $E_{K_{VH}}(r_V \| T_V \| E_{K_{MH}}(r_M \| K_{MH}))$ to H for identity authentication.

Step 3) After receiving the message from V, H first decrypts $E_{K_{VH}}(r_V \| T_V \| E_{K_{MH}}(r_M \| K_{MH}))$ by using K_{VH}. Then H determines whether the timestamp T_V is within the reasonable threshold compared with its current time. If it is not valid, H terminates the process. Otherwise, H gets the real identity of mobile user M by computing:

$$ID_M = PID_M \oplus h(N \| ID_H) \oplus ID_H \tag{3}$$

H then calculates the long-term key K_{MH} by $K_{MH} = f(ID_M)$ and uses K_{MH} to decrypt $E_{K_{MH}}(r_M \| K_{MH})$. If the decrypted secret key, K_{MH}, is equal to $f(ID_M)$, then M is authenticated. It also provides the implicit identity authentication of V. Subsequently, H sends $E_{K_{VH}}(r_V \| r_M \| h(ID_M))$ and $E_{K_{MH}}(r_M \| r_V \| ID_V)$ to V.

Step 4) Messages 4 and 5 show the process of the mutual authentication and key negotiation between M and V. On receiving the message from H, V first decrypts $E_{K_{VH}}(r_V \| r_M \| h(ID_M))$. If the decrypted random r_V^* is the same as its original random r_V, then V believes that M is an authorized user. Subsequently, V does the following: (1) Save the value $h(ID_M)$ for identifying the identity of user M in Phase II; (2) Set $K_{auth} = r_M \oplus r_V$ as the authentication key K_{auth}; (3) Forward message $E_{K_{MH}}(r_M \| r_V \| ID_V)$ to M.

Step 5) M decrypts $E_{K_{MH}}(r_M \| r_V \| ID_V)$ by using K_{MH}. If the decrypted random r_M^* is equal to the original r_M , then M computes the authentication key as $K_{auth} = r_M \oplus r_V$. Then, M sends $E_{K_{auth}}(K_{auth})$ to V to verify the key K_{auth}.

Step 6) If $D_{K_{auth}}(E_{K_{auth}}(K_{auth})) = K_{auth}$, then V records the authentication key K_{auth} for user M. So far, V has finished the authentication process with M and established an authentication key K_{auth}.

In the above steps, we illustrate the proposed authentication protocol I for secure roaming services. When M is staying in his home network, the authentication protocol for local services is shown in Fig. 2. Note that the difference between Fig. 1 and Fig. 2 is that the authentication protocol for local services ignores the original Messages 2 and 3 in Fig. 1.

Message 1. $M \rightarrow H$: $ID_H, PID_M, E_{K_{MH}}(r_M \parallel K_{MH})$
Message 2. $M \leftarrow H$: $E_{K_{MH}}(r_M \parallel r_H \parallel ID_H)$
Message 3. $M \rightarrow H$: $E_{K_{auth}}(K_{auth})$

Fig. 2. Authentication Protocol for Local Services Based on Secret Splitting Principle

Phase II: One-Time Session Key Progression

The main function in phase II is to establish and renew a session key between users *M* and *V*. In this phase, we introduce a novel mechanism called "One-time session key progression". The mechanism allows mobile *M* to renew his session key frequently and reduces the risk to use a compromised session key to communicate with *V*.

Suppose that mobile user *M* is required to renew his session key K_i with *V* for the i^{th} time, he can obtain the new session key K_{i+1} according to the steps shown in Fig. 3. $K_{i+1} = r_{M,i} \oplus r_{V,i}, i = 1,2,...,n$. Specially, K_1 is set as the authentication key K_{auth} (Phase I), i.e., $K_1 = K_{auth}$. The pseudonym identity $PID_{M,i}$ for user *M* is calculated as $PID_{M,i} = h(ID_M) \oplus r_{M,i}$, and hence it will vary in each session key negotiation because of the random number $r_{M,i}$.

Message 1. $M \rightarrow V$: $ID_V, PID_{M,i}, E_{K_i}(r_{M,i} \parallel K_i)$
Message 2. $M \leftarrow V$: $E_{K_i}(r_{M,i} \parallel r_{V,i} \parallel ID_V)$
Message 3. $M \rightarrow V$: $E_{K_{i+1}}(K_{i+1})$

Fig. 3. One-way Session Key Progression

As shown in Fig. 3, on receiving the message 1 from *M*, *V* can get the original random $r_{M,i}$ generated by user *M* by computing the following equation:

$$r_{M,i} = PID_{M,i} \oplus h(ID_M) = (h(ID_M) \oplus r_{M,i}) \oplus h(ID_M)) \tag{4}$$

V verifies whether the decrypted random $r^*_{M,i}$ is equal to the original one $r_{M,i}$. If it is true, *V* decrypts $E_{K_i}(r_{M,i} \parallel K_i)$ by using session key K_i and checks whether the decrypted session key K^*_i is the same as the session key K_i preserved by *V* in the previous key negotiation. If it is true, *V* terminates the execution. Otherwise, the identity of *M* is authenticated. Subsequently, *V* does the following: (1) Generate a random $r_{M,i}$; (2) Set $K_{i+1} = r_{M,i} \oplus r_{V,i}$ as the next session key and keep it; (3) Send $E_{K_i}(r_{M,i} \parallel r_{V,i} \parallel ID_V)$ to *M*.

Since random $r_{M,i}$ ($r_{V,i}$) can be known only by user *M* (*V*), K_i plays a role of one-time key. Therefore, the new mechanism is called as "One-time session key progression".

2.3 Anonymity and Intractability Analysis

The anonymity of user M is obtained by hash function and the smart card issued by his home network H. M hides his real identity in his pseudonym identity PID_M, i.e., $PID_M = h(N \| ID_H) \oplus ID_M \oplus ID_H$. Since only H knows the secret N, nobody except H can recover the real identity ID_M by computing $ID_M = PID_M \oplus h(N \| ID_H) \oplus ID_H$ (Step 3).

The intractability is achieved by two measures: (1) The $PID_{M,i}$ in each session key progression is different due to the random $r_{M,i}$; (2) The session key $K_{i+1} = r_{M,i} \oplus r_{V,i}$ is *one-time-use* so that there is no direct relationship between session keys. The random numbers guarantee the freshness of PID and session key in each session.

2.4 Attack Analysis

Firstly, we analyze the *co-operation attacks* in Phase II (Fig. 3). In Specific, there is domain separation between visited networks. When a user enters a new visited network, he will send a new different temporary identity $PID_{M,i}$ to the new visited network. Moreover, the session key K_{i+1} changes with the variation of the random number $r_{M,i}$ and $r_{V,i}$. So even though there is a co-operation between visited networks, a new visited network still cannot know the user's real identity.

Secondly, we consider the *impersonate attacks* in this protocol. (1) An intruder has no way to impersonate H, since he does not possess the long-term secret key K_{VH} and hence it is impossible for him to generate the responding confirmation $E_{K_{VH}}(r_V \| r_M \| h(ID_M))$ to V (in Step 3); (2) V also has no way to impersonate H to cheat user M, since the long-term key K_{MH} is unknown to V and he cannot generate $E_{K_{MH}}(r_M \| r_V \| ID_V)$ which contains the random r_M chosen by M.

Finally, we study the *relaying attacks*. In order to illegally obtaining an authentication key, an intruder attempts to impersonate a legal user by replaying the user's exchanged messages. He intercepts the Message 1 (step 1) sent by M and then replays Message 1 $\{ID_H, PID_M, E_{K_{IH}}(r_I \| K_{IH})\}$ to V, where $E_{K_{MH}}(r_M \| K_{MH})$ has been changed into $E_{K_{IH}}(r_I \| K_{IH})$. However, the intruder cannot get the correct message 3 from H, because the relation between PID_M and $E_{K_{MH}}(r_M \| K_{MH})$ in the original message 1 is self-encryption and can authenticate each other (step 3). Therefore, the proposed protocol is able to resist such replaying attacks.

3 Protocol II for Secure Roaming Services

The proposed protocol II is based on the self-certified scheme. This scheme combines the advantages of certificated-based and identity-based public key cryptosystems [11]. Regarding to the security strength of self-certified scheme, Saeednia [9] indicated that forging a valid witness w_i for user U_i is equivalent to break an instance of RSA cryptosystem.

The key idea of the proposed protocol is to consider home network *H* as a temporary TTP (Trusted Third Party) for roaming services. When *M* visits *V*, both of them initialize a registration procedure with *H* (*V* acts as an access agent for *M*). If *M* and *V* successfully register with *H*, they will obtain a witness from *H*, respectively, and the trust relations between *M* and *V* are established. In the consequent communications, *M* can directly negotiate the session key with *V* without accessing *H*. Similar to the proposed protocol I, this protocol also composes of two execution phases: Phase I) Mutual authentication protocol for registration; Phase II) Session key exchange protocol. The protocol uses public key algorithm and can be applied when visited network and home network do not have pre-setup shared secret.

Phase I: Mutual Authentication Protocol for Registration

User *M* chooses a random $r_M \in Z_u$ and computes $y_M = g^{r_M} \bmod (n)$ as his public key. Similarly, *V* also generates a random $r_V \in Z_u$ and calculates $y_V = g^{r_V} \bmod (n)$ as his public key. Next, let I_M and I_V be two strings associated with the personal information (Name, Address, etc.) of users *M* and *V*, respectively. In addition, suppose w_M and w_V be the witness of users *M* and *V*. Both are issued and calculated by *H* as follows:

$$w_M = ((y_M \oplus I_M)^{f(I_M)^{-1}}) \bmod (n), \tag{5}$$

$$w_V = ((y_V \oplus I_V)^{f(I_V)^{-1}}) \bmod (n). \tag{6}$$

Message 1. *M*→*V*: y_M, ID_H, TID_M
Message 2. *V*→*H*: $y_M, y_V, E_{K_{VH}}(y_V \| ID_V \| TID_M \| T_V)$
Message 3. *V*←*H*: $E_{K_{VH}}(w_V \| I_V), E_{K_{MH}}(w_M \| I_M \| ID_V)$
Message 4. *M*←*V*: $E_{K_{MH}}(w_M \| I_M \| ID_V)$

Fig. 4. Authentication Protocol Based on Self-Certified Scheme

Then the new authentication protocol for roaming services can be described in Fig. 4.

As shown in Fig. 4, the shared key K_{MH} are computed as $K_{MH} = (PK_H)^{r_M}$, where the random r_M is generated by *M* and the public key $PK_H = g^{SK_H}$ of *H* is already delivered to user *M* through a secure channel in advance. And the real identity ID_M of user *M* is hidden in the temporary identity TID_M, which is computed as $TID_M = E_{K_{MH}}(g^{r_M} \oplus ID_M)$.

We describe our proposed protocol II as follows.

Step 1) User *M* does the following: (1) Generate a random r_M and compute $y_M = g^{r_M}$; (2) Compute the shared key K_{MH} by $K_{MH} = (PK_H)^{r_M}$ and use it to compute $TID_M = E_{K_{MH}}(g^{r_M} \oplus ID_M)$; (3) Send ID_M, y_M and TID_M to *V*.

Step 2) V generates a random r_V, computes $y_V = g^{r_V}$, and sends y_M, y_V and $E_{K_{VH}}(y_V \parallel ID_V \parallel TID_M \parallel T_V)$ to H.

Step 3) H decrypts $E_{K_{VH}}(y_V \parallel ID_V \parallel TID_M \parallel T_V)$ by using shared key K_{VH}. If the timestamp T_V is reasonable and the decrypted value y_V^* is equal to clear-text y_V, H computes the shared key K_{MH} by $K_{MH} = (g^{r_M})^{SK_H}$, and decrypts the $TID_M = E_{K_{MH}}(g^{r_M} \oplus ID_M)$ by using K_{MH}. Then H can get the real identity of user M by computing the following formula:

$$ID_M = D_{K_{MH}}(E_{K_{MH}}(g^{r_M} \oplus ID_M)) \oplus g^{r_M}. \tag{7}$$

Then, H verifies the authenticity of ID_M. If it is legal, H (temporary TTP) does the following: (1) Prepare two strings I_M and I_V associated with the personal information of user M and V, respectively; (2) Compute the witnesses w_M and w_V for M and V (Eq. 5 and 6). (3) Send $E_{K_{VH}}(w_V \parallel I_V)$ and $E_{K_{MH}}(w_M \parallel I_M \parallel ID_V)$ to V.

Step 4) V decrypts $E_{K_{VH}}(w_V \parallel I_V)$ and verifies witness w_V and I_V by checking if

$$y_V = ((w_V)^{f(I^V)} \bmod(n) \oplus I_V). \tag{8}$$

If it is true, V successfully registers with H, and believes that M is an authorized user. V forwards $E_{K_{MH}}(w_M \parallel I_M \parallel ID_V)$ to M.

Step 5) Similarly, M decrypts $E_{K_{MH}}(w_M \parallel I_M \parallel ID_V)$ and verifies I_M and witness w_M by checking if

$$y_M = ((w_M)^{f(I_M)} \bmod(n) \oplus I_M). \tag{9}$$

If it is true, M successfully registers with H, and believes that the trust relations between M and V are also established with the assistance of home network H.

In addition, if user M is located in his home network, the authentication protocol can be described in Fig. 5.

Message 1. $M \rightarrow H$: y_M, ID_H, TID_M
Message 2. $M \leftarrow H$: $E_{K_{MH}}(w_M \parallel I_M \parallel ID_H)$

Fig. 5. Authentication Protocol for Local Services Based on Self-Certified Scheme

3.2 Phase II: Session Key Renewal Protocol

The one-time session key progression mechanism for this protocol is different from our previous protocol and the protocol in [4]. It renews the session key by utilizing a modified self-certified scheme and Diffie-Hellman mechanism (Fig. 6).

Message 1. $M \rightarrow V$: w_M, I_M, g^{t_M}
Message 2. $M \leftarrow V$: w_V, I_V, g^{t_V}

Fig. 6. Session Key Exchange Protocol

In Fig. 6, random $t_M, t_V \in Z_u^*$ denote two different elements of Z_u^* of order *u*. The key K_{MV} can be computed respectively by user *M* and *V* as follows.

For mobile user *M*, the procedure for acquiring the session key is

$$y_V = (w_V^{f(I_V)} \bmod(n)) \oplus I_V, \tag{10}$$

$$K_M \equiv (y_V^{t_M} \cdot (g^{t_V})^{r_M}) \bmod(n) = g^{r_V t_M + r_M t_V} \bmod(n), \tag{11}$$

$$K_{MV} \equiv h(K_M). \tag{12}$$

For visited Network *V*, the session key is acquired as:

$$y_M = (w_M^{f(I_M)} \bmod(n)) \oplus I_M, \tag{13}$$

$$K_V \equiv (y_M^{t_V} \cdot (g^{t_M})^{r_V}) \bmod(n) = g^{r_V t_M + r_M t_V} \bmod(n), \tag{14}$$

$$K_{MV} \equiv h(K_V). \tag{15}$$

The session keys calculated by *M* and *V*, respectively, are equal because

$$K_{MV} \equiv h(K_M) = h(g^{r_V t_M + r_M t_V} \bmod(n)) = h(K_V), \tag{16}$$

where *h* is a collision-resistant hash function. Key confirmation is done implicitly during the session. Moreover, this protocol can yield a different key for each session.

The security of the key exchange is especially enhanced by using the protocol, since every session key is used for one time. Compared with the previous protocols, we obtain two extra properties: (1) Decreased the number of message exchanges to two; (2) One-time session key progression mechanism.

3.3 Anonymity Analysis

As shown in Fig. 4, the real identity ID_M of user *M* is hidden in his temporary identity TID_M, which is computed as $TID_M = E_{K_{MH}}(g^{r_M} \oplus ID_M)$, where $K_{MH} = (PK_H)^{r_M}$. On the other hand, since only *H* knows its secret key SK_H, nobody except *H* can calculate K_{MH} and decrypt the TID_M. Therefore, *H* can get the real identity of user *M* according to Eq. 7, which is another mechanism for identity anonymity.

3.4 Attack Analysis

Firstly, consider the *relaying attacks* in session key renewal protocol (Fig. 6) such that an adversary pretends to act as *M* and tries to exchange a secret key with *V*, who intends to share the secret key with *M*. The adversary can randomly choose an integer $\alpha \in Z_u^*$; then he sets $r_M^* = \alpha \cdot f(I_M)$ as a fake secret key for *M* and replace *M*'s original public key y_M with $y_M^* = g^{r_M^*} \bmod(n)$. However, the adversary cannot compute a valid witness w_M^* for *M*, because the original witness $w_M = ((y_M \oplus I_M)^{f(I_M)^{-1}}) \bmod(n)$ for user *M* is self-certified. Therefore, although the adversary can intercept the message $\{w_M, I_M, g^{t_M}\}$, he still cannot forge the correct message $\{w_M^*, I_M, g^{t_M}\}$ which satisfies the following relation: $w_M^* = ((y_M^* \oplus I_M)^{f(I_M)^{-1}}) \bmod(n)$, unless he can compute discrete logarithm modulo a large composite [7]. So it can be seen that the proposed protocol is able to resist such replaying attacks, i.e., the adversary and *V* cannot obtain the same secret key. Similarly, an adversary that impersonates *V* also cannot obtain the same secret key with user *M*.

Secondly, consider the *impersonation attacks* in the Mutual Authentication Protocol (Fig. 4). (1) An intruder has no way to impersonate *H* since he does not possess the long-term secret key K_{VH}, and hence it is impossible for him to generate the responding confirmation $E_{K_{VH}}(w_V \| I_V)$ to *V*; (2) *V* has no way to impersonate *H* to cheat user *M*, since the long-term key K_{MH} is unknown to *V*, and *V* cannot generate $E_{K_{MH}}(w_M \| I_M \| ID_V)$ where w_M contains y_M generated by *M*.

4 Performance Analysis

The performance comparisons between the two proposed protocols and the protocol in [4] are shown in Tables I and II, in which Phase I and Phase II of these three protocols are described, respectively. We mainly compare the number of hash operation, symmetric encryption/decryption, exponential operation, and the number of message exchanges. Note that the rows in shade show the comparisons related to mobile user *M*.

From to Tables I and II, we can generally conclude that though the identity anonymity mechanism is introduced into our protocols for the roaming service, the complexity of our protocols is no more than that of the protocol in [4], and the computation requirement for mobile device is quite low.

In addition, the proposed protocol II reduces the number of symmetric encryption/decryption operations, and increases the exponentiation operations. Though the exponentiation is a relatively time-consuming operation, some exponentiations can be pre-computed, e.g., $g^{r_M}, g^{t_M}, g^{r_V}$, and g^{t_V}. Hence, the real exponentiation computation load is not remarkable. The protocol II also provides: (1) identity anonymity; (2) the mutual authentication between the two entities without pre-setup shared secret key; (3) the session keys are generated for each session. All the features are especially favorable and safer in the roaming environment. The computational load increase resulting from the identity anonymity and one-time session key progression provides the enhanced security that are not available in the protocol in [4].

Note that the exponentiation operations required for *M* is in Eq. 9 (Phase I) and Eq. 11 (Phase II), respectively. If we only consider the exponentiation operations except those pre-computed exponentiation operations, the average computation complexity is $\frac{3}{2} \cdot \left\lfloor \log(\frac{n}{2}) \right\rfloor \cdot M(n)$, where $M(n)$ denote the computation complexities of modular modulo *n*. Actually, according to the binary algorithm for fast exponentiation [12], computing g^x will take $2 \cdot \lfloor \log x \rfloor$ multipliers in the worst case, and $\frac{3}{2} \cdot \lfloor \log x \rfloor$ on the average. So the complexity of computing Eq. 9 and 11 can be approximately as

Table I. Performance Comparisons (Phase I)

Comparison Item		Protocol in [4]	Proposed I	Proposed II: Self-certified
Exponential operation	M	N/A	N/A	1+2 Pre.
	V	N/A	N/A	1+1 Pre.
	H	N/A	N/A	3
Hash operation	M	1 (step 1)	N/A	1 (step 1)
	V	N/A	N/A	N/A
	H	1 (step 3)	1 (step 3)	1 (step 3)
Symmetric Encryption	M	2 (step 1, 5)	2 (step 1, 5)	1 (step 1)
	V	1 (step 2)	1 (step 2)	1 (step 2)
	H	2 (step 3)	2 (step 5)	2 (step 5)
Symmetric Decryption	M	1 (step 5)	1 (step 5)	1 (step 5)
	V	2 (step 4, 6)	2 (step 4, 6)	1 (step 4)
	H	2 (step 3)	2 (step 3)	1 (step 3)
Transmissions	$M \leftrightarrow V$	3	3	2
	$V \leftrightarrow H$	2	2	2
Anonymity		N/A	Yes	Yes

Table II. Performance Comparisons (Phase II)

Comparison Item		Protocol in [4]	Proposed protocol I	Proposed II: Self-certified
Exponential operation	M	N/A	N/A	1+2Pre
	V	N/A	N/A	1+2Pre
Symmetric encryption	M	1	1	N/A
	V	1	1	N/A
Symmetric decryption	M	1	1	N/A
	V	1	1	N/A
Transmissions	$M \leftrightarrow V$	3	3	2
Anonymity		N/A	Yes	Yes

M (Mobile); *V* (Visited Network); *H* (Home Network); Pre (Pre-computation exponentiation)

$\frac{3}{2}\cdot\left\lfloor \log(\frac{n}{2})\right\rfloor$ on the average. In Eq. 11, the exponentiation operation for $y_V^{t_M} \bmod(n)$ can be pre-computed while $(g^{t_V})^{r_M} \bmod(n)$ cannot be computed in advance since the random t_V is only determined by *V*, and varies in every session key renewal phase.

5 Conclusions

Two novel mutual authentication and key exchange protocols with identity anonymity and one-time session key progression are proposed for GLOMONET. For each protocol, identity anonymity has been achieved by hiding the real user identity in prearranged PIDs based on the secret-splitting principle or by encrypting the real identity with the shared key, respectively. The proposed protocols can protect a mobile user's privacy in the roaming network environment and reduce the risk that a mobile user uses a compromised session key to communicate with visited networks. The two protocols can be applied depending on the availability of the long-term shared secret key shared by the home network and its users. The performance comparisons have shown that the complexity of our protocols is similar to the protocol in [4] with significant security improvement.

Acknowledgement

This research work was supported in part by the National Natural Science Foundation of China under contracts No.90104002, 60273009, 60373013 and 60372019, and 60218003; the Projects of Development Plan of the State High Technology Research under contract No. 2003CB314804, and Intel IXA University Research Plan (China Research Center).

References

1. S. Suzukiz and K. Nakada, "An authentication technique based on distributed security management for the global mobility network," *IEEE Journal on Selected Areas in Communications*, vol. 15, issue 8, pp. 1606-1617, 1997.
2. Z. J. Tzeng and W. G. Tzeng, "Authentication of mobile users in third generation mobile system," *Wireless Personal Communication*, vol. 16, issue 2, pp. 35-50, 2002.
3. L. Buttyan, C. Gbaguidi, and et al., "Extensions to an authentication technique proposed for the global mobility network," *IEEE Trans. on Communication*, vol. 48, issue 3, pp. 373-376, 2000.
4. K. F. Hwang and C. C. Chang, "A self-encryption mechanism for authentication of roaming and teleconference services," *IEEE Trans. on Wireless Communications*, vol. 2, issue 2, pp. 400-407, 2003.
5. G. Horn and B. Preneel, "Authentication and payment in future mobile system," *Computer Security - ESORICS '98, LNCS*, vol. 1485, pp. 277-293, 1998.
6. D. S. Wong and A. H. Chan, "Mutual authentication and key exchange for low power wireless communications," *Proc. of IEEE Military Communications Conference, MILCOM 2001*, vol. 1, pp. 39-43, 2001.
7. S. L. Ng and C. Mitchell, "Comments on mutual authentication and key exchange protocols for low power wireless communications," *IEEE Communications Letters*, vol. 8, issue 4, pp. 262-263, 2004.
8. S. Saeednia, "Identity-based and Self-certified Key Exchange Protocols," *Proc. of the Second Australian Conference on Information Security and Privacy*, pp. 303-313, 1997.
9. S. Saeednia, "A note on Girault's self-certified model," *Information Processing Letters, Elsiver*, vol. 86, issue 6, pp. 323-327, 2003.
10. B. Schneier, "Applied cryptography: protocols, algorithm, and source code C," *John Wiely & Sons, Inc (Second Edition)*, pp. 70-72, 1996.
11. M. Girault, "Self-certified public keys," *Advance in Cryptology - Eurocrypt '91*, pp. 491-497, 1991.
12. R. L. Adelman and K. S. McCurley, "Open problem in number theoretic complexity," *Proc. of the 1994 Algorithmic Number Theory Symposium, Springer-verlag*, pp. 291-322, 1994.

An Energy-Efficient Image Representation for Secure Mobile Systems

Tim Woo, Catherine Gebotys, and Sagar Naik

Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, Ontario, Canada
timwoo@alumni.uwaterloo.ca, cgebotys@uwaterloo.ca,
knaik@swen.uwaterloo.ca

Abstract. In a mobile device, two major sources of energy consumption are energy used for computation and energy used for transmission. Computation energy can be reduced by minimizing the time spent on compression. Transmission energy and encryption energy can be reduced by sending a smaller image file that is obtained by compression. Image quality is often sacrificed in the compression process. Therefore, users should have the flexibility to control the image quality to determine whether such a tradeoff is acceptable. This paper proposes an energy efficient image representation system using Binary Space Partitioning (BSP) trees. Our experimental result shows that in most cases, our new tree construction and compression formula use only 40% of the original total energy required for compressing and sending images. BSP tree representation also allows partial encryption, which reduces total energy required for secure transmission by reducing the amount of data that needs to be encrypted. Our experimental results show that BSP tree representation is effective in reducing total energy for secure transmission for computer arts images compared to JPEG.

1 Introduction

The increasing popularity of wireless digital image communication presents a new challenge to image compression and encryption. First, portable devices run on a limited energy source such as batteries. Therefore, energy utilization must be efficient. From a software perspective, there are several strategies for optimizing energy [1]. These strategies focus on minimizing the energy cost for computations and memory accesses. For a wireless device, the total energy consumption consists of energy used for computations, data accesses, and transmission energy. For example, a typical CPU for PDA (whose name is not revealed to protect vendor identity) consumes about 90-150mW in running mode, 36mW in idle mode, and 0.9mW in sleeping mode. The transceiver consumes about 210-540mW for transmitting and 180-240mW for receiving data. Since the power used for computation and transmission is quite significant, this paper focuses on reducing the computation energy and transmission energy. To minimize computation energy, one can reduce the execution time of the compression algorithm, so that less

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 126–137, 2005.

energy is consumed in the switching activities in the processor when computing. To reduce transmission energy, one can reduce the amount of data to be sent, which is achieved by compression.

Although the study of image quality and execution time tradeoff is not a novel idea, it deserves special attention for a wireless device because of its limited computation power, memory and energy source. At the time of writing, a typical PC consists of a 2.8GHz processor, with 512MB RAM. A typical PDA (e.g. PalmOne Tungsten T3) consists of a 400MHz processor, with 64MB RAM, which is only a fraction of the computation power and memory available to a PC. Therefore, it is especially crucial for the image compression and encryption system to be optimized for a wireless PDA.

To provide flexibility and high compression performance, a Binary Space Partitioning (BSP) tree structure can be used to represent an image [2]. BSP tree representation is easy to construct, and it allows convenient pruning to different image quality levels. In a BSP tree, each non-terminal node is associated with a partition line and each leaf node is associated to a particular region of the image. The non-terminal nodes store the line parameters while the leaf nodes store the colour of the associated regions.

We have proposed a new tree construction technique by splitting the image into two equal halves each time instead of finding the optimal line. We have found in our experiments that the original Least-Square-Error (LSE) method in [3] is slow. We have also proposed a new pruning formula for compressing images. In BSP lossy compression, calculation of errors at each node is required to make pruning decisions. Our new formula aims to reduce computation times by reusing previous calculated values in previous iterations. Finally, we have introduced the concept of variable compression rate for different areas of the image using the advantages of BSP.

This paper is organized as follows: Sect. 2 discusses the relevant previous research. Sect. 3 describes the proposed BSP tree construction and pruning formula. Sect. 4 shows the experimental results by varying different parameters. Sect. 5 analyzes the new proposed scheme and discusses how our results compare to previous work, followed by a summary of the advantages and limitations of the new scheme and the concluding remarks in Sect. 6.

2 Related Work

There have been several studies on the effect of varying JPEG parameters on the image quality, latency, and energy. In [4], the authors have used a pre-computed lookup table to estimate compresssion costs. Research in [5] has studied the effect of varying JPEG parameters on the image quality. Although their approach does not require complete decompression to re-scale an image to different quality, it still requires re-encoding of entropy parameters, which consumes significant energy too. In contrast, our BSP tree compression scheme consumes only a small amount of energy when re-scaling because no re-computations are necessary when pruning the tree. JPEG encryption [6] has revealed that there are many image leakages when only part of the JPEG file is encrypted. For example, the image is still recognizable even if most of the AC and DC coefficients are encrypted.

A recent standard, JPEG2000, uses wavelet transforms to compress an image. In wavelet compression [7], the image first goes through a discrete wavelet transform (DWT) to generate wavelet coefficients. In [8], the wavelet coefficients are organized in pyramid levels according to importance, with each level adding additional details to the image. The resulting image consists of two parts: the zerotree structure and the wavelet coefficients.

Several energy efficient wavelet image compression schemes have been proposed. In [9],[10], the authors have proposed varying parameters such as the transform level, elimination level (EL), and the quantization level (QL). Again, the authors have used a lookup table that requires re-computation of quantization and entropy coding steps, which consumes fair amount of energy too. In [11], a distributed approach to wavelet compression has been proposed to distribute energy use in a wireless adhoc network evenly at different nodes. In such a system, the task for the wavelet transform, is processed in a distributed fashion to the various nodes in the adhoc network to save energy.

Quadtree compression [12] is similar to BSP except an image is split into four quadrants instead at each iteration. The advantage of quadtree compression is that tree construction is fast and simple. However, since each split creates four quadrants, it may result in a larger file size compared to BSP.

Many of the image compression algorithms are designed to aim for the maximum compression rate. However, these high compression algorithms may be too complex for energy constrained handheld devices. BSP scheme has advantages that other image compression techniques do not have. First, the hierarchical structure of the BSP tree allows convenient pruning of an image. This offers scalability and flexibility by dynamically adjusting the image quality according to the user need and environment to save transmission energy. In addition, since BSP compression operates in spatial domain, it is possible to compress using different thresholds in different regions of the same image. This allows selectively retaining the details in the important areas of an image, while the less important areas can be compressed more.

3 Our Image Representation Scheme

This section presents the proposed scalable energy efficient image representation scheme using BSP trees. We present the specifics of how the BSP tree representation of the highest quality image is constructed, and the specifics of how the energy, and image quality parameters are used in different modules of the system.

Fig. 1 shows the block diagram for our image compression system. The end user specifies his desired security levels and image quality for the various regions of the image (to be used for selective thresholding) to the system. From these user parameters, the test bench module generates the threshold for the compression

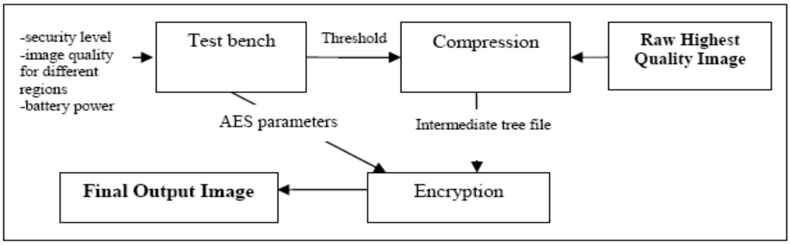


Fig. 1. Our Image Compression and Encryption System

and the encryption parameters. The compression module then compresses the highest quality image and the resulting image is immediately encrypted using Advanced Encryption Standard (AES) to generate an encrypted image. Only the defining BSP tree needs to be encrypted for security (not the colours defining each leaf node of the BSP tree).

In our proposed binary split method, the algorithm simply bisects the longest dimension of the region of interest in two equal halves. This speeds up the line selection process significantly, since it eliminates the expensive LSE transform computation in [3]. However, the trade-off is that the file size may not be optimal. In our method, for a region R with endpoints p_1 and p_2 and x_1 and x_2 as the x-coordinates (y-coordinates if it is a vertical line) of the two points, the midpoint between x_1 and x_2, i.e. x_C, is simply selected as the split line. The image is partitioned into smaller rectangular regions until the colour of the region is homogenous or until the region cannot be splitted further. Although this splitting method violates the property of optimal alignment of BSP tree partitioning lines with the actual object edges, it is found in our experiments that, in practice, this new method still performs satisfactorily.

To compress highest quality image file in BSP tree form, we can prune the BSP tree according to the pruning threshold, which is expressed as a fraction relative to the root node's error in our experiments. This process reduces the file size by sacrificing the image quality. There exist some advanced pruning techniques such as [13]. For simplicity, we focus our work using a simple error based pruning criterion.

Before compression, the error associated with each node in the tree must be calculated. We now introduce the concept of pruning. Pruning deletes and merges nodes in a BSP tree if the error is smaller than a user specified threshold. In the best quality image, the leaf nodes store the exact colours of the original image. When pruning this best quality tree, these leaf nodes are merged to form new leaf nodes if the merged node's error is below the pruning threshold, which is expressed as a fraction of the root node's error in our work. After these leaf nodes are merged, the mean colour value of the pixels in the region is used to approximate the new leaf node's colour. First, users will specify a relative threshold divisor value. Then, this value is multiplied by the root node's error value to generate the absolute threshold value that is used for pruning decision.

Traditionally, error for the region is calculated using the total square error. However, in our experiment, it is found that this formula is slow. Therefore, we have proposed a new error calculation formula for image pruning. Our formula calculates error in a bottom-up manner and speeds up computation by reusing previously calculated error values. First, at each iteration i, two parameters, σ_1 and σ_2, are defined:

$$\begin{aligned}\sigma_1 &= \begin{cases} |m_L - m_i| & \text{if left child of the current node is a leaf;} \\ \sigma_L & \text{otherwise} \end{cases} \\ \sigma_2 &= \begin{cases} |m_R - m_i| & \text{if right child of the current node is a leaf;} \\ \sigma_R & \text{otherwise} \end{cases}\end{aligned} \tag{1}$$

where m_L and m_R are the mean values of the left and right child, respectively, σ_L and σ_R are the previously calculated values of σ for the left and right child, respectively and m_i is the mean value of the current node at iteration i. The current node's standard deviation σ_i is calculated by:

$$\sigma_i = \begin{cases} 0 & \text{if left child of the current node is a leaf;} \\ \sqrt{\frac{A_L\sigma_1^2 + A_R\sigma_2^2}{A_L + A_R}} & \text{otherwise} \end{cases} \tag{2}$$

where A_L and A_R are the area of the left and right child, respectively. Finally, the node's total error E_i is calculated by this formula:

$$E_i = \begin{cases} A_L\,|m_L + \sigma_1 - m_i| + A_R\,|m_i - (m_R - \sigma_2)| & \text{if } m_L \geq m_i; \\ A_R\,|m_R + \sigma_2 - m_i| + A_L\,|m_i - (m_L - \sigma_1)| & \text{otherwise} \end{cases} \tag{3}$$

After the image is compressed by BSP tree algorithm, the image goes to encryption stage. In the encryption module, only the BSP tree part needs to be encrypted. Unlike JPEG, where the entire image must be encrypted for secure transmission, BSP tree allows partial encryption while still keeping the image secure. This saves encryption computation energy cost.

4 Experimental Results

The experimental results of varying different parameters in generating an image will be presented in this section. We will investigate the effect of the tree construction mechanism on the computation energy and compression bit rate. We will also investigate how the error formula used for pruning affects the image quality and compression ratios. The computation energy is estimated by multiplying the execution time by the estimated average CPU power dissipation during execution. Due to limited resources, the execution time is measured on a PC, and the time on PDA is estimated by scaling down by a factor of seven (since a typical PC is 2.8GHz and a typical PDA CPU is 400MHz). Then the result is multiplied by 120mW, the average CPU power consumption during computation. The transmission time is estimated from the file size of the image multiplied by the average data rate of 600kbit/s for a typical 3G system [14]. Then the transmission energy is estimated by multiplying the estimated transmission time by 375mW, the average measured transmission power consumption of the transceiver for our PDA.

Table 1. Tree Construction and Transmission Energy Consumption (Joules) for Various Images. The total energy is the sum of the computation energy required for constructing the tree and the transmission energy for sending the image

	Optimal Line			Binary Split			$\frac{E_{\mathrm{Total}_\mathrm{B}}}{E_{\mathrm{Total}_\mathrm{O}}}$
	E_{tree}	E_{Tx}	$E_{\mathrm{Total}_\mathrm{O}}$	E_{tree}	E_{Tx}	$E_{\mathrm{Total}_\mathrm{B}}$	
peppers (512 × 512)	11.86	5.23	17.09	1.30	5.58	6.88	0.40
lena_small (256 × 256)	2.74	1.33	4.07	0.30	1.37	1.67	0.41
tulips (768 × 512)	17.47	7.96	25.43	1.95	8.46	10.41	0.40
frymire (512 × 512)	8.85	1.46	10.31	0.80	3.32	4.12	0.40
serrano (512 × 512)	9.67	1.12	10.79	0.66	2.33	2.99	0.27

4.1 Effect of Tree Construction Mechanism

The tree construction method affects how fast the best quality tree representation of image can be generated. In addition, the shape of the tree affects the pruning characteristics. This section presents the resulting images for the optimal line selection method and the binary split method.

Table 1 shows the computation energy E_{tree} used for tree construction and the transmission energy E_{Tx} for both the original LSE optimal line and our binary split method. It can be seen that our method only uses about 40% of the original total energy required for constructing and sending the image. Most of the savings comes from the savings in the computation energy for tree construction. Although the images generated by our method use slightly more transmission energy because the generated file is slightly larger, it is compensated by the much more significant savings in computation.

4.2 Effect of Error Calculation Formula on Pruning

In this experiment, we will explore how our error pruning formula performs on the optimal line constructed and our binary split constructed tree. We will explore the effects of the tree construction method on the results generated by our pruning formula. The proposed fast error formula in (3) is used to prune optimal-line tree and binary-split tree images at a threshold of 4.00×10^{-4}. The resulting peppers and frymire images are shown in Fig. 2.

With respect to image quality, it is found that our fast error formula works slightly better with binary split generated trees. The optimal line generated image's quality degrades quickly as the threshold increases. For example, at a threshold

Table 2. Computation Energy Consumption (Joules) for Computing Node Error

	Optimal Line Tree		Binary Split Tree	
	TSE	Our Error Formula	TSE	Our Error Formula
Peppers	4.99	0.50	3.77	0.53
Frymire	1.08	0.14	2.19	0.33

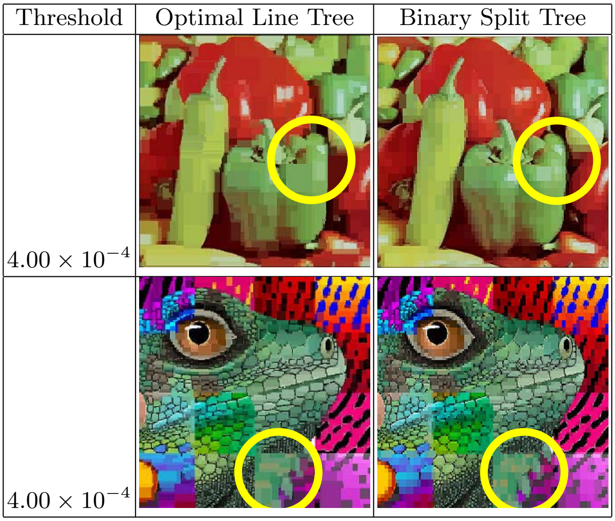


Fig. 2. Peppers and Frymire Images Using Fast Error Pruning. For peppers image, our fast error formula works better images trees constructed with binary split method, since the tree is more balanced. For frymire, our pruning formula works equally well with image trees constructed with either the optimal line or our binary split method

of 5×10^{-6} of the root node's error, the bit rate decreases quickly to 3.49bbp at 34.54dB, while the binary split tree has a slower drop in bit rate (8.82bbp) and image quality (37.73dB). The sharp drop in bit rate and image quality for optimal line trees is due to the observation that these trees are usually unbalanced. Therefore, the effect of error propagation is worse for an unbalanced tree where some paths to the root may be very long. As a result, the root's total error calculated using our formula deviates from the actual total square error more. Therefore, our formula performs better for our binary split constructed trees.

The estimated computation energy consumed in computing the errors at each node is shown in Table 2. For both the peppers and frymire images, the computation energy is reduced by almost 90%. Also, the slow total square error (TSE) method works slightly better on binary split tree version than the optimal line tree version of peppers. Again, this is due to the unbalanced tree shape of optimal line trees. Therefore, a lot of the calculations done at the deeper levels need to be performed again. For the frymire image, the behaviour is opposite. The binary split tree version is slower. This is because for the frymire image, the best quality image actually has a lot more nodes than the optimal line version.

Although the optimal line selection algorithm yields an optimal size image tree, it is too computationally intensive; therefore it is not optimal for energy. On the other hand, the binary split method can construct the BSP tree much faster, with a slight increase in file size. Moreover, calculating the total square error in the traditional way is too slow. Our proposed a fast error formula yields satisfactory results when pruning images, especially for binary split method constructed images.

4.3 Selective Focus Regions Feature

One of the major advantages of a BSP tree is the feature of selectively choosing important areas of the image to have a higher quality than the less important parts. The "importance" of a region is supplied as user parameters. For the less important areas, one can choose a higher threshold to compress the image more, while in the more important areas one can choose a lower threshold to keep a higher quality image. For example, in Fig. 3a, the lena image is pruned with the same threshold for the entire picture. The facial features and the background are pruned with the same quality. As a result, the facial features are blurred (highlighted with circles). On the other hand, in Fig. 3b, the facial region is pruned with a lower threshold, while the less important background is pruned with a higher threshold. One can see that Fig. 3b appears to have a higher quality than Fig. 3a because the important facial details are conserved. Selective focus region feature should theoretically work well for quadtrees too, although it has not been suggested in previous research in [12].

5 Discussion

This section presents the analysis for the various characteristics of BSP tree representation. We will analyze the performance of the tree construction methods and pruning formula, image quality with respect to execution time and file size, and the encryption performance with respect to file size and block or key length.

In [3], LSE is calculated for every potential partitioning line. The minimum total number of operations required for constructing a optimal line selection BSP tree for a $N \times N$ image is $MinTotalOps = 2\log_2 N \cdot (3N^2 \cdot Multiplications + 5N^2 \cdot Additions)$. This is only the lower bound of the total number of operations required for the optimal line selection process. Usually, more than one line passes the LSE transform condition. Therefore, the number of actual computations is significantly greater.

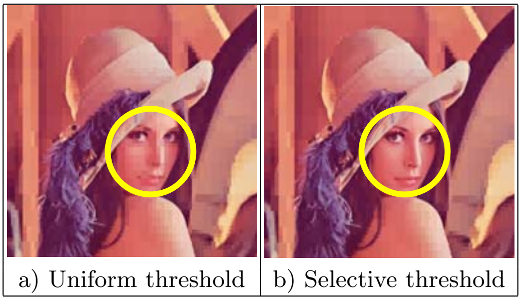

a) Uniform threshold b) Selective threshold

Fig. 3. Lena Image with Selective Threshold Regions. It can be seen that with selective threshold to preserve the facial details, the lena image is visually higher quality than the same image with uniform threshold across the image

In the binary split case, for each split, only one addition and one multiplication is needed to find the midpoint. Thus, only $N^2 - 1$ additions and $N^2 - 1$ multiplications are required. No calculation of total square error is needed. It can be seen that the binary split method is much faster than the optimal line selection method. From the experimental results, it can be seen that the file size increase due to neglecting the optimal line is acceptable. Therefore, the binary split method is a simple yet efficient method to construct the BSP tree. It has the combined benefits of the efficiency of representation of a BSP tree and the efficiency in tree construction of a quadtree.

In [3], TSE needs to be calculated at every level of the tree. This may yield to better pruning decisions, yet it is not efficient since this calculation needs to be repeated again at the higher levels. Our proposed formula reuses the partial error previously calculated from the lower nodes. This speeds up the pruning exponentially. Calculation of the total square error requires N^2. In contrast, our proposed formula in (3) reuses the children's total error and derives the new total error by adding a weighted average of the children node's error and standard deviation. No recalculation of error for the subtree regions is needed.

Our proposed algorithm aims to reduce energy consumption by cutting down the computation energy for constructing the image tree and pruning the tree. In the new binary split tree construction method, the tree construction energy is cut down significantly, at the expense of a slight increase in tree file size and transmission energy. In the fast error calculation method, the computation energy is reduced by reusing results from previous calculations. The tradeoff is a slightly less accurate pruned image quality.

In general, the higher the image quality, the larger the file size is. Table 3 shows the file size, bit rates and the PSNR for the peppers and frymire images compared to JPEG. The BSP tree with the bit rate closest to the best compression (denoted BC) JPEG is chosen for comparison. The optimal tree peppers image is pruned at 10^{-5} threshold using the original TSE formula. The best quality image (denoted BQ) chosen for comparison with the JPEG is also generated by optimal tree method.

The BSP tree with the bit rate closest to the best compression JPEG is chosen for comparison. The optimal tree frymire image pruned at 10^{-4} threshold. The best quality image for comparison with the JPEG is the optimal tree image.

Table 3. BSP Tree and JPEG Transmission Energy and PSNR

	JPEG size (bytes)	$E_{\mathrm{Tx_{JPEG}}}$ (Joules)	PSNR (dB)	BSP size (bytes)	$E_{\mathrm{Tx_{BSP}}}$ (Joules)	PSNR (dB)
Peppers BC	23308	0.117	32.19	34020	0.170	25.29
Peppers BQ	249787	1.249	44.35	1116072	5.580	56.53
Frymire BC	88012	0.440	21.13	86006	0.433	20.39
Frymire BQ	475277	2.376	47.49	200632	1.014	49.66

It can be seen that BSP tree compression works better with the frymire image than the peppers image. For the frymire image, the BSP tree compression can compress more than 2 times better at the best quality (BQ) settings in Table 3. Even if we set JPEG at the best compression (BC) settings, BSP tree compression is still able to match the compression ratio of JPEG. Table 4 illustrates the estimated total energy required for encrypting and transmitting the JPEG and BSP images. For JPEG it is assumed that the entire JPEG file is encrypted to avoid security leaks. In the BSP case only the defining BSP tree is encrypted as described in Sect. 3. It can be seen that our BSP approach uses only a fraction of the encryption energy, denoted E_{AES}, compared to JPEG. When sending the frymire image, our BSP approach is better than JPEG, and it uses only 38-87% of the total energy required for encrypting and transmitting an equivalent JPEG image. However, it uses more energy than JPEG for the peppers image. From our experimental results, it appears that BSP trees works better for computer graphics images and JPEG is more suitable for natural photographic images.

We have compared the bit rate with other compression systems. For the data based on 8-bit greyscale images, we have multiplied the bit rate by 3 for a fair comparison to the 24-bit colour used in this system. Table 5 shows the comparison to other image compression systems. The values in parentheses are the greyscale values multiplied by 3. Our BSP tree compression system is comparable to other systems' performance for the peppers image. However, for the lena image, other systems work better. Unfortunately, we are unable to find the data for other images from the literature.

Our proposed BSP compression algorithm is faster than [3] because it reuses calculations at previous nodes. With respect to the compression performance and based upon our experimental results, BSP trees are better suited for computer graphics images than photographic images when compared to JPEG. This is because BSP compresses better when there are larger homogenous regions, which

Table 4. Estimated BSP Tree and JPEG Secure Transmission Energy (Joules)

	JPEG			BSP Tree			$\frac{E_{\mathrm{Total}_{\mathrm{BSP}}}}{E_{\mathrm{Total}_{\mathrm{JPEG}}}}$
	E_{AES}	E_{Tx}	$E_{\mathrm{Total}_{\mathrm{JPEG}}}$	E_{AES}	E_{Tx}	$E_{\mathrm{Total}_{\mathrm{BSP}}}$	
Peppers BC	0.015	0.117	0.132	0.001	0.170	0.17	129%
Peppers BQ	0.164	1.249	1.413	0.041	5.580	5.621	398%
Frymire BC	0.058	0.440	0.498	0.003	0.430	0.433	87%
Frymire BQ	0.311	2.376	2.689	0.011	1.003	1.014	38%

Table 5. Compression performance compared to other systems

	Quadtree		JPEG		Fractal Binary Tree		BSP Tree	
	Bit rate	PSNR	Bit rate	PSNR	Bit Rate	PSNR	Bit Rate	PSNR
peppers	0.516	N/A	N/A	N/A	0.60 (1.80)	32.5	1.67	31.37
lena	0.67 (2.01)	30.36	0.125 (0.375)	26.2	0.24 (0.73)	31.0	0.73	26.57
frymire	N/A	N/A	1.813	47.49	N/A	N/A	0.765	49.66

is typical for computer graphics. In contrast, photographic images have smooth transitions between pixels. Therefore, neighbouring pixels are usually slightly different from each other. JPEG is more specialized in handling such cases. Our BSP scheme is also better than quadtrees in most cases because partitioning lines are more flexible, thus less nodes are created to represent the same image.

Finally, this paper has proposed the idea of allowing users to select a different pruning threshold for different regions using a BSP tree. This allows users to conserve the details in the important parts of the image. After all, it is the end users' perception of the image that is ultimately important. Compression algorithms that affect the image globally (e.g. JPEG and wavelets) do not allow such flexibility. We have proposed a binary split scheme in building the BSP tree, which combines the benefits of both worlds: it has the computation efficiency of quadtree in splitting the image, but also has the compact storage property of an optimal line BSP tree.

6 Conclusion

In this paper, we have presented an energy efficient image representation scheme. For mobile systems, a fast compression algorithm is necessary to save computation costs and the energy cost used for transmission. Our binary split tree construction scheme combines the benefits of both quadtree and optimal line BSP tree: it has the execution efficiency of quadtree in splitting the image, but also has the compact storage property of an optimal line BSP tree. Our experiment results show that the file size, hence, transmission energy, only increases slightly when using a binary split scheme. Our BSP tree pruning formula uses less energy than the traditional total square error formula because it reduces computations by reusing results from previous calculations. With respect to compression performance, we have found that BSP tree compresses computer graphics better than photographic images. Since computer graphics have large homogenous regions, our BSP compression algorithm is effective in merging those neighbouring pixels together. Finally, our image representation allows users to choose different pruning thresholds for different areas of the image. By choosing a higher threshold for the less important areas, while maintaining a lower threshold for the important areas, the impact on the perception of image quality is small. At the same time, transmission energy is reduced because the image file size is smaller.

The authors would like to thank A. Sung for his power measurements.

References

1. Naik, S., Wei, D.: Software implementation strategies for power-conscious systems. ACM Mobile Networks and Applications **6** (2001) 291–305
2. Radha, H., Vetterli, M., Leoonardi, R.: Image compression using binary space partitioning trees. IEEE Trans. on Image Processing **5** (1996) 1610–1624

3. Radha, H., Vetterli, M., Leoonardi, R.: Fast piecewise constant approximation of images. SPIE Visual Comm. and Image Processing **1605** (1991) 475–486
4. Chandra, S., Ellis, C.: JPEG compression metric as a quality aware image transcoding. In: Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, CO (1999) 81–92
5. Taylor, C., Dey, S., Panigrahi, D.: Energy/Latency/Image quality tradeoffs in enabling mobile multimedia communication. Software Radio - Technologies and Services (2001) 55–66
6. Kailasanathan, C., Safavi Naini, R.: Compression performance of JPEG encryption scheme. In: IEEE Intl Conf. on Digital Signal Processing. Volume 2. (2002) 1329–1332
7. Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. IEEE Trans. on Image Processing **41** (1993) 3445–3462
8. Said, A., Pearlman, W.A.: A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans. on Circuits and Systems for Video Technology **6** (1996) 243–250
9. Lee, D., Dey, S.: Adaptive and energy efficient wavelet image compression for mobile multimedia data services. In: Proc. of Intl Conf. on Communications. Volume 4., New York (2002) 2484–2490
10. D.G.Lee, D.Panigrahi, S.Dey: Network-aware image data shaping for low-latency and energy-efficient data services over the Palm wireless network. World Wireless Congress (3G Wireless) (2003)
11. Wu, H., Abouzeid, A.: Energy efficient distributed JPEG2000 image compression in multihop wireless networks. In: Proc. of the 4th Workshop on Applications and Services in Wireless Networks, Boston, MA (2004)
12. Cheng, H., Li, X.: Partial encryption of compressed images and videos. IEEE Trans. on Signal Processing **48** (2000) 2439–2451
13. Chou, P., Lookabaugh, T., Gray, R.: Optimal pruning with applications to tree-structured source coding and modeling. IEEE Trans. on Information Theory **35** (1989) 299–315
14. Eyuboglu, V.: CDMA2000 1xEV-DO delivers 3G Wireless (2002) Network World, Feb 25th 2002, http://www.nwfusion.com/news/tech/2002/0225tech.html.

Efficient Clustering for Multicast Key Distribution in MANETs

Mohamed Salah Bouassida[1], Isabelle Chrisment[1], and Olivier Festor[2]

[1] LORIA-UHP
[2] LORIA-INRIA,
MADYNES, Campus scientifique,
B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex - France
name.surname@loria.fr

Abstract. Securing multicast communications in ad hoc networks must consider several challenging factors such as high nodes mobility, limited bandwidth and constrained energy. Moreover, the establishment of a key management protocol within ad hoc environments meets the "1 affects n" problem, which is more critical in such type of networks, due to the high dynamicity of groups.

In this paper, we present an efficient clustering scheme for multicast key distribution in Mobile ad hoc networks. This scheme divides the multicast group into clusters, according to the localization of the group members and their mobility. Simulations indicate a valuable reduction in the average latency of keys distribution and a promising reduction in energy consumption. Analysis also shows that our scheme is scalable.

Keywords: ad hoc networks, multicast security, group key management, energy-efficient, mobility-aware.

1 Introduction and Motivations

The Multicast transmission is an efficient communication mechanism for group oriented applications, such as video conferencing, interactive multi-party games and software distribution. One main advantage of multicast communication is to save networks resources, mainly by reducing the consumption of bandwidth and source and routers resources.

In parallel to the development of the multicast services within the Internet, the last decade saw the exponential deployment of ad hoc networks, thanks to the emergence of new wireless technologies and standards (e.g. 802.11, Hiperlan [6], ...). A mobile ad hoc network is an autonomous and dynamic collection of devices, that are connected without any fixed infrastructure, that can be highly mobile. This mobility implies that network topology changes rapidly and unpredictably and the connectivity among the hosts varies with time. Generally, mobile ad hoc networks operate on low power devices, having limited energy and bandwidth, low CPU process capability and small memory sizes.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 138–153, 2005.

The combination of an ad hoc environment with multicast services to be operated, induces new challenges towards the security infrastructure needed to enable acceptance and wide deployment of multicast communication.

To prevent attacks and eavesdropping, several services need to be provided such as authentication, data integrity and data confidentiality. The most suitable solution to provide these services is the establishment of a key management protocol. This protocol is responsible for the generation and distribution of the traffic encryption key (TEK) to all group members, to encrypt multicast data by the source and decrypt it by the receivers using this TEK, which ensures data confidentiality ; so that only authenticated members are able to receive the multicast flow sent by the group source.

Multicast key distribution has to take into account a challenging element which is the "1 affects n" phenomenon [8] (after a Join or a Leave procedure, the TEK is renewed and redistributed to all group members to keep forward and backward secrecy). To face this problem and to attenuate its impact on the protocol performances, several approaches propose a multicast group clustering [2, 3, 1, 8]. The clustering consists in dividing the multicast group into sub-groups. Each sub-group is managed by a local controller (LC), responsible for local key management within its cluster. Thus, after Join or Leave procedures, only members within the concerned cluster are affected by the rekeying process.

The "1 affects n" phenomenon is more critical in ad hoc networks due to the high nodes dynamicity and additional factors such as mobility awareness, energy and bandwidth consumption.

In this paper, we present a clustering scheme for multicast key distribution in mobile ad hoc networks, consisting in forming strongly correlated clusters, based on localization and mobility information of the group nodes. Our approach, delivers fast, energy-efficient and mobility-aware key distribution in a multicast service within MANETs, and aims to:

- exploit the advantage of broadcast communications in a wireless environment with omni-directional diffusion ;
- optimize the data transmission time. A correlated cluster minimizes the number of relays responsible for forwarding the messages emitted by the source of the group and consequently minimizes the data-transmission time ;
- optimize the bandwidth consumption. This advantage is induced by the minimization of the multi-hop retransmissions number within the cluster ;
- optimize energy consumption. The reduction of energy consumption in an ad hoc network is a true challenge because of the limitation of the batteries resources. A strongly correlated cluster should minimize energy consumption required for the transmission of a message to all members of the cluster. Indeed, this energy is equal to the sum of energy units used by each relay to retransmit the data to its downstream members. The minimization of these relays will decrease the total energy consumption.

To present our approach, the remainder of our paper is structured as follows. Section 2 presents the related work, concerning multicast key management approaches and energy-efficient multicast tree building. In section 3, we describe our clustering scheme for multicast key distribution. In section 4 and 5, we show the interest of clustering for energy reduction and we assess by simulation the efficiency of our approach. And finally, section 6 concludes this paper.

2 Related Work

2.1 Multicast Key Management Approaches

Several key management protocols for securing multicast communications have been elaborated over the last few years. We classified them into three main approaches (Figure 1): without clustering approach, with static clustering approach and with dynamic clustering approach.

Without clustering, all group members share a unique symmetric key (TEK), used within a centralized architecture where a single server is responsible for TEK generation and distribution. LKH (Logical Key Hierarchy) [14] belongs to this approach. In LKH, group members are organized as leaves of a tree with logical internal nodes. The root of the tree is the group key. The cost of a compromise recovery operation in LKH is proportional to the depth of the compromised member in the LKH tree. LKH scheme proposes maintaining a balanced tree, which gives a uniform cost of O(log n) for rekeying process in group of n members. The original LKH does not consider nodes mobility and localization in an ad hoc environment. The proposal in [7] enhances the distribution scheme of LKH within ad hoc networks by optimizing energy consumption with the use of geographical localization information related to members. The basic idea of this approach is that members who are geographically close to each other, can potentially be reached by a broadcast, or can use the same path to receive data. By representing group members in a two-dimensions space, [7] uses the K-means clustering algorithm to form groups with strong correlation, and deduces the multicast key distribution tree. But, the mobility of nodes is not considered. The design of multicast key management trees that match the cellular networks topologies is proposed in [11]. It reduces the communication cost by 55% to 80% compared to key management trees that are independent of the network topol-

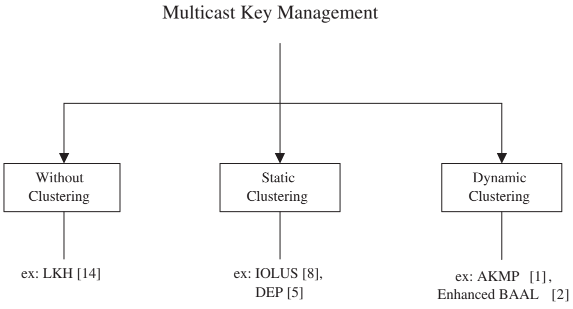


Fig. 1. Classification of multicast key management approaches

ogy. However, this scheme is built on the infrastructure of the cellular networks and is hardly applicable in ad hoc networks.

In the static clustering approach, the multicast group is initially divided into several subgroups. Each of them shares a local session key managed by a local controller. Protocols proposed within this approach like IOLUS [8] and DEP [5] are more scalable than centralized protocols. In IOLUS, each cluster holds its local TEK ; thus, the multicast flow should be decrypted and re-encrypted whenever it passes from a cluster to another. However, DEP achieves a better compromise between scalability and CPU processing by encrypting data with only one TEK which is shared by all clusters. Thus, local controllers have to decrypt and re-encrypt only the TEK and not all the multicast flow.

The dynamic clustering approach aims to solve the “1 affects n” problem without generating an overhead due to decryption/re-encryption process, this approach merges the two previous approaches. The basic idea is to start a multicast session with centralized key management (first approach), and to divide the group dynamically in order to delegate key management to the local controllers (second approach). Evaluation functions are proposed to decide when group clustering is achieved. AKMP [1], SAKM [4] belong to this approach and are dedicated to wired networks. whereas Enhanced BAAL [2] proposes a dynamic clustering scheme for multicast key distribution in ad hoc networks.

2.2 Energy-Efficiency Approaches for Multicast Trees Establishment

The energy efficiency is a challenging issue for multicast key distribution tree in an ad hoc environment. Some interesting research works were published on this subject.

The broadcast incremental power algorithm BIP [12] determines the minimum-power broadcast tree, rooted at the source node, and that reaches every node in the wireless network, by computing the best point of attachment, according to the minimum required energy to reach them from the source. This algorithm takes advantage of the broadcast within wireless environment, but does not consider node mobility. [12] deduces from BIP, an algorithm called Multicast Incremental Power (MIP), which computes the best multicast tree, always by optimizing the energy required to reach every group member from the source, but by pruning nodes which are not members of the group.

BLU (Broadcast Least-Unicast-Cost) [12] superposes the best unicast paths to each destination individually, according to the minimum cost metric. Neither the node mobility nor the advantage of the broadcast within ad hoc environment are considered. An approach dedicated to multicast communications, MLU (Multicast Least-Unicast-Cost) is almost identical to BLU with the difference that unicast paths are established only for the group members. Multicast tree will be built by superposing these theoretically optimal paths.

The third algorithm presented by [12] is BLiMST (Broadcast Link-Based MST), establishes a minimum-energy diffusion tree, using MST technique (Minimum-cost Spanning Tree). The wireless broadcast advantage is also ignored in this algorithm. An algorithm called MLiMST deduces a multicast tree from the one built by BLiMST, by pruning nodes which are not group members.

[12] affirms that MIP establishes the best multicast tree, within an ad hoc environment, having the minimum number of relays responsible for multicast flow forwarding, and the minimum required energy to reach all the group members.

3 Energy-Efficient and Mobility-Aware Clustering Scheme for Multicast Key Distribution

We present in this section a new dynamic clustering scheme for multicast key distribution dedicated to ad hoc networks. This scheme wants to optimize energy consumption and latency for key delivery. Being mobility aware, our approach needs the geographical location information of all the group members in the construction of the key distribution tree. Thus, we assume that within an ad hoc network, a *Global Positioning System* (GPS) is available.

Our framework network is defined with the following parameters:

1. Nodes are represented by points in a two-dimensions euclidian space, where distance between two nodes $i(x_i, y_i)$ and $j(x_j, y_j)$ is classicaly computed as follows:

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = \sqrt{\| i - j \|^2}$$

2. The required power for sending a message from a node i to node j is proportional to the distance $d(i, j)$, and it is computed as following [10]:

$$P = (d_{ij})^\gamma$$

γ is the propagation loss factor in the network that typically holds a value between 2 and 4, depending on the characteristics of the communication medium. We note that the application domain and context of our approach is heterogeneous ad hoc networks, which implies that each node in the network has its specific maximal range, depending on its physical characteristics.

3. Given a cluster with c local members, the distance matrix between theses members is D:

$$D = \begin{pmatrix} 0 \; d_{12} & \cdots & \cdots & d_{1c} \\ 0 \; \ddots & & & \vdots \\ \vdots & d_{ii} = 0 & & \vdots \\ \vdots & & \ddots & d_{c-1,c} \\ 0 \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

During multicast group initialization, every group member is attached to the group source, called global controller (GC). This entity is responsible for the TEK generation and its distribution to all group receivers. In addition, the GC verifies periodically whether the group is highly correlated, and consequently whether the key distribution process is optimal. The evaluation of the cluster cohesion is determined with a cluster cohesion parameter that we have defined as the centralization index of the cluster around the LC node (node 1):

$$Cohesion = \frac{members\_in\_the\_LC\_range}{cluster\_members\_number}$$

This parameter measures the proximity of the cluster members compared to their controller. The bigger the number of members reachable by the controller in one hop is, the closer the factor of cohesion reaches 1. With this cohesion parameter, we can verify if a cluster is strongly correlated or not.

We define $Min\_Cohesion$ as the minimum threshold that a cohesion factor of a cluster should not exceed. Otherwise, the controller must take the decision to split the cluster, and the election of new local controllers LCs according to their localization, must be initiated.

The split process optimizes the number of new LCs while reaching every member of the cluster. This process is done by the OMCT algorithm (Optimized Multicast Cluster Tree), whose principle is as follows (cf Algorithm 1):

- STEP 1: List_nodes is the list of the c cluster members. List_LCs is initially empty and will contain the local controllers of the new clusters which are built. Node 1 corresponds to the initial local controller of the cluster.
- STEP 2: Sort the list of members in the ascending order, according to their distances compared to the running LC, initially node 1.
- STEP 3: Traverse the list of members until the node $j + 1$ which cannot be joined directly by the LC. All of the j members which can be reached directly by current LC are withdrawn from the List_nodes.
- STEP 4: The nearest node to the node $j + 1$, belonging to the range of the running LC, will be elected as current LC. It is thus withdrawn of the List_nodes and added to the List_LCs.
- STEP 5: Iterate steps 2, 3 and 4 until having processed all the cluster members. Thus, all the members are reachable by the List_LCs and can start receiving multicast flow sent by the source.

Once the first clusters are created within the multicast group, the new LCs become responsible for the local key management and distribution to their local members, and also for the maintenance of the strongly correlated clusters property.

Thus, recursively, and at every moment of a multicast group session, the group is composed of strongly correlated clusters, ensuring that their respective cohesion is always higher than the definite minimal threshold $Min\_Cohesion$.

Algorithm 1. OMCT: Optimized Multicast Cluster Tree

```
List_LCs = $\phi$ (STEP1)
List_nodes = {1, 2, 3, ..., c}
i = 1;
j = 2;

while (List_nodes $\neq \phi$) do
  Sort_List_nodes(i) ; // Sort the list of members according to their distances compared to the node i (STEP2)
  while $d_{(i,List\_nodes(j)} \leq Max\_Distance)$ do
    // Max_Distance corresponds to the range of the node i
    $List\_nodes = List\_nodes/\{List\_nodes(j)\}$ ; // Remove List_nodes(j) of the members list
    j++;
  end while
  //The node List_nodes(j+1) cannot be reached directly by the node i (STEP3)

  s=1;
  for ($l = 2$ to $j$) do
    if ($d_{(s,List\_nodes(j+1)} > d_{(l,List\_nodes(j+1))}$) then
      s=l;
    end if
  end for
  LC=s;
  // Elected LC will be closest to the node List_nodes(j+1) (STEP4)
  i=LC;
  $List\_LCs = List\_LCs \cup \{LC\}$ ; // Add LC to the local controllers list
end while(STEP5)
```

Figure 2 illustrates an execution example of this algorithm. The new LCs are nodes 1, 4 and 8.

The OMCT algorithm 1 achieves a group clustering, according to the localization and the mobility of members group. However, the number of members of each new cluster is not taken into account. Thus, in the example of the Figure 3, node 2 forms a cluster with only one member (node 3). Whereas node 4 can reach node 3, without forming a new cluster. An improvement of the OMCT algorithm is then essential, considering the number of members of each new cluster. This number is moving between two thresholds (cf Algorithm 2). The principals enhancements of the algorithm are as follows:

- STEP 3': Traverse the nodes list, sorted in ascending order according to their distances compared to the current LC. Nodes in the range of the current LC are withdrawn from the List_nodes and added to the list of the local members of this LC, while respecting the constraint of maximum threshold of local numbers (MAX_THRESHOLD). At the end of this step, the node $j + 1$ cannot be reached directly by the current LC, or the cluster managed by this LC is saturated.

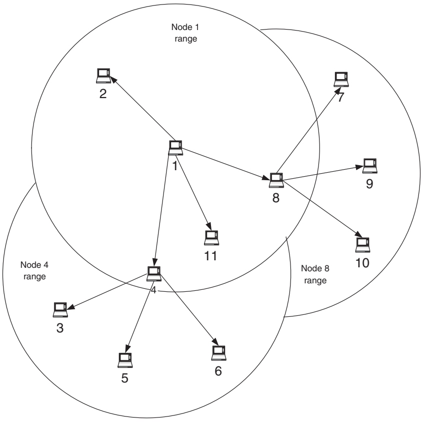


Fig. 2. Illustration example of OMCT

- STEP 6': Traverse the formed clusters. If a cluster has a local members number lower than the minimal threshold MIN_THRESHOLD, then traverse the local members of this cluster and try to move them to others clusters, as far as possible.

Mobility Management. Periodically, each LC computes the cohesion parameter of its cluster. According to the value of this parameter, it decides or not to execute the OMCT algorithm to clusterize its sub-group.

When a member moves within the network, it can be no more reachable by its LC. All LCs send periodically "LC_Queries" messages containing their identities and their coordinates. Thus, a member moving in the network, and

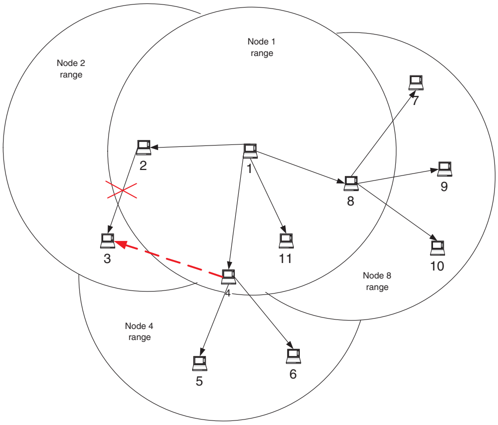


Fig. 3. Illustration example of OMCT V2

Algorithm 2.OMCT: Optimized Multicast Cluster Tree V2

```
List_LCs = $\phi$ (STEP1)
List_nodes = {1, 2, 3, ..., c}
i = 1;
j = 2;
nb_LCs = 0;

// Initialization of all the local members lists
for ($m$ = 1 to $c$) do
  $List\_local\_members\_\{m\} = \phi$
end for
while (List_nodes $\neq \phi$) do

  Sort_List_nodes(i) ; // Sort the list of members according to their distances compared to the
  node i (STEP2)

  while $d_{(i,List\_nodes(j)} \leq Max\_Distance$) do
    if ($nb\_List\_local\_members\_\{i\} < MAX\_THRESHOLD$) then
      $List\_nodes = List\_nodes/\{List\_nodes(j)\}$ ; // Remove List_nodes(j) of the members list
      $Liste\_membres\_locaux\_\{i\} = Liste\_local\_members\_\{i\} \cup \{List\_nodes(j)\}$;
      j++;
    else
      Break;
    end if
  end while
  //The node List_nodes(j+1) cannot be reached directly by the node i or the cluster is already
  saturated (STEP3')

  s=1;
  for ($l$ = 2 to List_nodes(j)) do

    if ($d_{(s,List\_nodes(j+1)} > d_{(l,List\_nodes(j+1))}$) then
      s=l;
    end if
  end for
  LC=s;
  // Elected LC will be closest to the node j+1 (STEP4)
  i=LC;
  $List\_LCs = List\_LCs \cup \{LC\}$ ; // Add LC to the local controllers list
  nb_LCs ++;
end while(STEP5)

// Algorithm improvement holding account of the members number of new clusters (STEP6')
for ($t$ = 1 to $nb\_LCs$) do
  if ($nb\_List\_local\_members\_\{t\} < MIN\_THRESHOLD$) then
    for ($m$ = 1 to $nb\_List\_local\_members\_\{t\}$) do
      for ( $z = t$ to $nb\_LCs$ ) do
        if ($d_{List\_LCs\{z\},m} \leq Max\_Distance$ & $nb\_List\_local\_members\_\{z\} <$
        $MAX\_THRESHOLD$) then

          $List\_local\_members\_\{t\} = List\_local\_members\_\{t\}/\{m\}$ ;
          // Remove m of the local members list of t
          $List\_local\_members\_\{z\} = List\_local\_members\_\{z\} \cup \{m\}$;
          // Add m of the local members list of z
          Break;

        end if
      end for
    end for
  end if
end for
```

receiving LC_Queries from LCs, chooses to join the nearest LC, according to its localization. Figure 4 illustrates this mobility scenario.

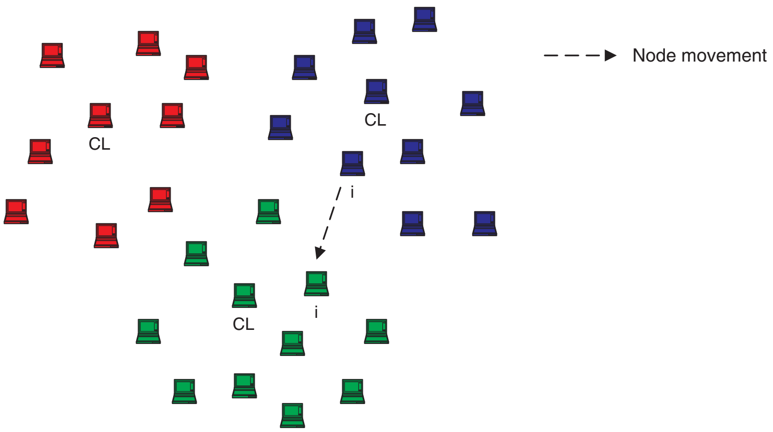


Fig. 4. Nodes mobility

Then, for its re-authentication and access control to the new LC, this member uses the concept of the re-authentication ticket. This concept requires that local controllers form a restricted multicast group and share a session key called KEK_{LCs}. The re-authentication ticket contains a password encrypted with the KEK_{LCs}, and is sent securely to the member, during its first authentication in the multicast group. After receiving the ticket, the new LC will decrypt it, check the password and in case of success, the member will be linked to the new cluster. The advantage of the ticket is to allow the LCs to decide about re-authentication and access control of the new members without needing to ask the global controller for it. We note that the ticket must be updated by LCs, periodically, after each member revocation within the multicast group, and when a LC leaves the group. This fact is necessary and mandatory in the case of revocation of a group member which should not be able to re-join the group, since any cluster within the group. A member who left the group for a duration longer than the ticket update period, cannot be re-authenticated using its ticket.

When a LC moves in the network, leaves the group or disappears due to any resources problems, it must previously send a notification message to all its local members asking them for moving to others clusters having nearest LCs. Then, these LCs will run, if necessary, the OMCT algorithm.

4 Analysis and Discussions

This section presents the analysis realized to validate our cohesion parameter with OMCT algorithm, within ad hoc networks. The topologies are generated randomly and the nodes localization is randomly distributed on all of the network surface. The OMCT algorithm is executed with 10 different topologies. A node is randomly selected to be the multicast group source. Others nodes are selected to be members of the group and the remaining nodes are simple participants.

We choose two variable factors which are group members number (10-20-50) and network surface (50*50 -100*100 - 150*150). The number of nodes within the network is fixed at 100. Propagation loss factor is chosen as $\gamma = 2$, without loss of generality. We assume a maximum range of 50 m. We note that radio frequency of 2,4 Ghz in 802.11b offers a rather important range which can vary from 30 to 100 m, even 400 m following the used hardware.

Our discussion is based on the following metrics:

- the number of new LCs corresponding to the number of new clusters.
- the required energy to send a message in multicast by the source to all of group members. Only the energy required for message sending is computed, the energy used at message reception and that used at signals processing are neglected.
- we use also a metric presented by [13], called *yardstick metric*:

$$yardstick = \left(\frac{m}{p}\right)\left(\frac{m}{n}\right) = \left(\frac{m^2}{n * p}\right)$$

n is the destinations number (group members number), m is the group members number reached by the multicast flow sent by the source, and p is the sum of energies necessary for sending a multicast message. (m/n) represents the attainability factor, and (m/p) represents the energy-consumption factor per reached members number. More optimal is the clustering approach, greater is the yardstick metric.

For the performed analysis, the thresholds of members number within a cluster are not considered. We note also that the local controllers are selected only if they are group members, because they hold the traffic encryption key and can reach the multicast flow. Moreover, all the local controllers are ad hoc nodes and not dedicated routers as wired networks, and it is understandable that a node which is not a group member, does not ensure the role of local controller for this group.

The table in figure 5 shows the number of new clusters compared to the cohesion of the initial group. Smaller the cohesion is, more we need to create new clusters within our multicast group. The creation of new clusters implies the creation of new LCs used as relays to their local members, thus requiring more transmission energy. The table 5 shows also the relationship between the yardstick metric and the group cohesion factor. When the cohesion of the group increases, the local controller can reach more group members while minimizing the required energy. That means the reachability factor (m/n) and the energy-consumption factor (m/p) increase, and consequently the yardstick metric.

We note that the number of created clusters is only influenced by the cohesion factor and not by the number of group members. This implies that our OMCT clustering scheme is scalable. We can also validate our cohesion factor, because it is proportional to the required energy, the number of new clusters

	Surface	Cohesion	Clusters Number	Required Energy	Yardstick Metric
n = 10	50 * 50	0.99	1.1	1571.73	6.36
	100 * 100	0.5	2.5	3716.45	2.13
	150 * 150	0.35	2.8	4628.53	0.97
n = 20	50 * 50	0.985	1.2	1908.82	10.60
	100 * 100	0.445	3.2	5420.88	3.24
	150 * 150	0.32	3.9	6286.40	1.90
n = 50	50 * 50	0.982	1.2	2195.48	23.03
	100 * 100	0.522	3.9	6749.87	7.47
	150 * 150	0.376	4.9	9229.56	4.34

Fig. 5. Clusters Number, Required Energy and Yardstick Metric vs Cohesion

and the yardstick metric. From this cohesion factor, we will deduce the minimum threshold, below which a multicast group must run the OMCT clustering algorithm.

To compare our OMCT clustering algorithm with the MIP approach [12], we use the same example presented in [12]. With the same topology, OMCT produces a cluster with cohesion equal to 1, using only one local controller which is the node source 10 (cf Figure 6) ; this solution is optimal in terms of energy consumption and key distribution latency. Whereas the MIP algorithm provides a key distribution scheme requiring more relays and more consumption of energy.

The proposal of [7] presented in section 2.1, uses a refined K-means algorithm, which enhances the Logical Key Hierarchy (LKH) in energy efficiency. K-means procedure provides 15% to 37% savings from the worst possible assignment of the LKH approach. However, as said in [7], there are cases where the application of K-means induces higher energy consumption than a randomly generated tree. This fact revealing the non optimality of the approach in some cases, comes from the fact that K-means algorithm fails to exploit the broadcast advantage in a wireless environment. Infact, in the specific example of two nodes which are at the same distance from a LC but in an opposite direction, K-means fails to clusterize them together, whereas, with the OMCT algorithm, the two nodes will belong to the same cluster, thus benefits from the broadcast advantage, saving energy and decreasing average key distribution latency in the group.

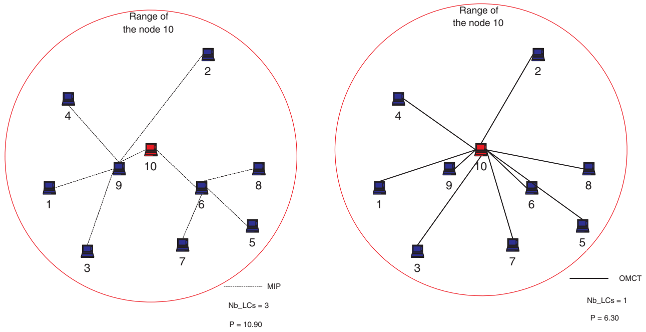


Fig. 6. OMCT vs MIP

5 Simulations and Results

The objective of our simulations is to compare the OMCT-clustering approach with the non-clustering and the localization-independent clustering approaches, according to the following factors:

- Latency: average delay for keys delivery from a sender to a receiver,
- Energy: sum of energy units required for messages transmission throughout the simulation duration.

The non-clustering approach consists in having a centralized multicast group, with one controller responsible for TEK distribution periodically to all group members. The forward and backward secrecy are ensured within this approach triggering a TEK redistribution process after each Join or Leave procedure.

The localization-independent clustering approach that we use in our simulations belongs to dynamic clustering protocols family. The clustering is achieved dynamically, by applying a simple evaluation function whose parameters are the number of the local group members and the local dynamicity frequency (number of Joins and Leaves per unit of time). Each group member computes its evaluation function. If the number of the local group members or the local dynamicity frequency exceeds a defined threshold, a member creates with its local members a cluster and becomes a local controller. Within this approach, the TEK is distributed periodically by the local controllers to all their group members. In addition, each LC distributes a local cluster key to its local members, after each Join or Leave procedure. This clustering approach is presented by [2].

We run several simulations under Linux Mandrake 10.0, using the network simulator NS2 version ns-allinone-2.26. The simulation environment is composed of:

- area: 500*500 meters.
- number of nodes 50 - 100.
- simulation duration: 1000s.
- physical/Mac layer: IEEE 802.11 at 2Mbps, 250 meters transmission range.
- mobility model: random waypoint model with no pause time, and mode movement speed 0m/s, 1m/s and 10m/s.
- multicast routing protocol is MAODV [9]. [15] presents MAODV implementation under NS2.26.
- the member arrival follows a Poisson process (λ arrival per time unit) and the membership duration is an exponential distribution with average μ.
- the group source is node 0.
- only distribution keys traffic exists in the simulation.

Figure 7 shows the average latency for keys delivery, using the non-clustering approach, the localization-independent clustering approach and the OMCT-clustering approach. Our OMCT-clustering approach brings a profit in latency computed as 28% to 31% compared to the non-clustering approach, and 15% to 27% compared to the localization-independent clustering approach. Indeed, the

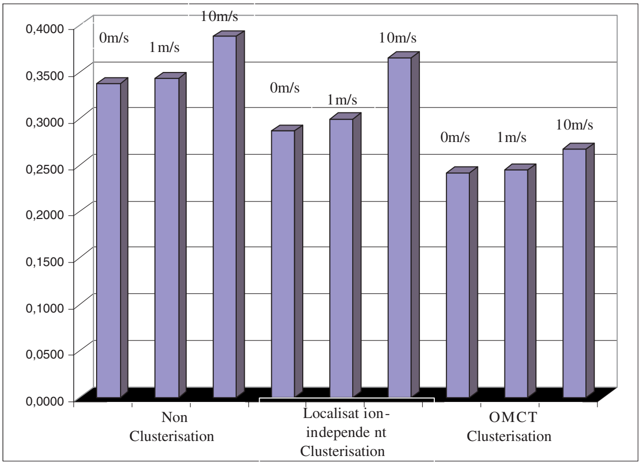


Fig. 7. Latency

non-clustering approach suffers from the "1 affects n phenomenon", the unique LC is responsible for the TEK distribution to every group member, after each Join or Leave procedure. Group members are also randomly localized in the ad hoc network. Thus, the delay for keys delivery increases. The latency computed for the localization-independent clustering and the OMCT-clustering approach is smaller than latency within non-clustering approach, due to their attenuation of the "1 affects n" phenomenon. However, the most optimal latency is computed within our OMCT-clustering approach because it ensures strongly correlated

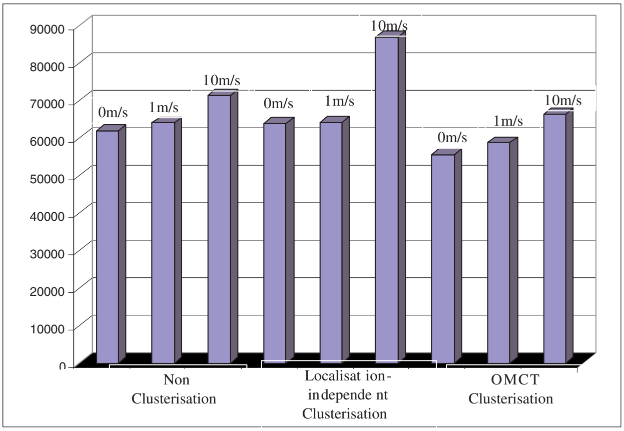


Fig. 8. Energy

clusters, each cluster being managed by a LC close to its local members which optimize valuably the keys delivery delay.

Figure 8 indicates that OMCT brings also a profit concerning the required energy for keys distribution within the multicast group, computed as 7% to 10% compared to the non-clustering approach, and 8% to 23% compared to the localization-independent clustering approach.

The energy-efficiency can be explained as the latency optimization. Indeed, while forming a strongly correlated clusters within the OMCT-clustering approach, having a cohesion factor close to 1, the local controllers have to consume less energy to reach every member in the cluster.

However, the mobility factor affects the performances of our OMCT approach. This is due to the time necessary for the construction of the clusters. However, even in a high mobility environment, OMCT remains the most optimal concerning energy efficiency and average of latency.

6 Conclusion and Future Works

In this paper, we presented an efficient clustering scheme for key distribution dedicated to ad hoc networks. Our approach based on OMCT algorithm, is mobility-aware and energy-efficient. We divide the multicast group dynamically, into clusters, according to the localization and mobility of the group members. Each cluster is managed by a local controller responsible for the local key management.

Periodically, each LC verifies whether its cluster is highly correlated and consequently whether the local key distribution is optimal within its subgroup, by computing an evaluation parameter called cluster cohesion. This parameter measures the proximity of the cluster members compared to their controller, so that formed clusters are strongly correlated, thus optimizing the average latency of key distribution and the sum of energy units required for multicast messages transmission.

The OMCT algorithm will be integrated within a key management protocol approach in ad hoc networks, called BALADE [3]. This protocol delivers a fast and efficient key distribution in a sequential multi-sources multicast service, that we developed within our research team.

As future works, we will analyse the performance of our algorithms in terms of time complexity, while integrating the probability theory in these analyses. we plan also to enhance our scheme by integrating the key distribution reliability which is an important issue in ad hoc networks due to the high packets-loss rate in such environment.

References

1. H. Bettahar, A. Bouabdallah, and Y. Challal. An adaptive key management protocol for secure multicast. In *11th International Conference on Computer Communications and Networks ICCCN*, Florida USA, October 2002.

2. M.S. Bouassida, I. Chrisment, and O. Festor. An Enhanced Hybrid Key Management Protocol for Secure Multicast in Ad Hoc Networks. In N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, and L. Merakos, editors, *Networking 2004, Third International IFIP TC6 Networking Conference*, volume 3042 of *Lecture Notes in Computer Science LNCS*, pages 725–742, Athens, Greece, May 9-14 2004. Springer.
3. M.S. Bouassida, A. Lahmadi, I. Chrisment, and O. Festor. Diffusion multicast sécurisée dans un environnement ad-hoc (1 vers n séquentiel). Rapport de recherche, INRIA, Sep 2004.
4. Y. Challal, H. Bettahar, and A. Bouabdallah. SAKM: A Scalable and Adaptive Key Management Approach for Multicast Communications. *ACM SIGCOMM Computer Communications Review*, 34(2), April 2004.
5. L. Dondeti, S. Mukherjee, and A. Samal. Secure one-to-many group communication sing dual encryption. In *Comput. Com-mun.23, 17 (November)*, 1999.
6. B. Jones and D. Skellern. Hiperlan channel assignment strategies. In *Electronics Letters*, 1997.
7. L. Lazos and R. Poovendram. Energy-Aware Secure Multicast Communication in Ad Hoc Networks Using Geographical Location Information. In *IEEE International Conference on Acoustics Speech and Signal Processing*, 2003.
8. S. Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM*, pages 277–288, 1997.
9. E. Royer and C. Perkins. Multicast Ad hoc On-Demand Distance Vector (MAODV) routing, IETF Internet Draft: draft-ietf-manet-maodv-00.txt, 2000.
10. J. Sadowsky and V. Kafedziski. On the Correlation and Scattering Functions of the WSSUS Channel for Mobile Communications. *IEEE Transactions On Vehicular Technology*, 47(1), February 1998.
11. Y. Sun, W. Trappe, and K. Liu. A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks. *IEEE/ACM Transactions on Networking*, 12(4):653–666, August 2004.
12. Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks. In *INFOCOM 2000*, Tel-Aviv, Israel, March 26–30 2000.
13. Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. Algorithms for energy-efficient multicasting in static ad hoc wireless networks. *Mobile Networks and Applications*, 6(3):251–263, 2001.
14. Chung K Wong, Mohamed G. Gouda, and Simon S. Lam. Secure group communications using key graphs. Technical report, University of Texas at Austin, 1997.
15. Y. Zhu and T. Kunz. Maodv implementation for ns-2.26 - systems and computing engineering, carleton university, technical report sce-04-01, January 2004.

Using Secure Coprocessors to Protect Access to Enterprise Networks

Haidong Xia, Jayashree Kanchana, and José Carlos Brustoloni

Dept. Computer Science, University of Pittsburgh,
210 S. Bouquet St. #6135, Pittsburgh, PA 15260, USA
{hdxia, kanchana, jcb}@cs.pitt.edu

Abstract. Enterprise firewalls can be easily circumvented, e.g. by attack agents aboard infected mobile computers or telecommuters' computers, or by attackers exploiting rogue access points or modems. Techniques that prevent connection to enterprise networks of nodes whose configuration does not conform to enterprise policies could greatly reduce such vulnerabilities. Network Admission Control (NAC) and Network Access Protection (NAP) are recent industrial initiatives to achieve such policy enforcement. However, as currently specified, NAC and NAP assume that users are not malicious. We propose novel techniques using secure coprocessors to protect access to enterprise networks. Experiments demonstrate that the proposed techniques are effective against malicious users and have acceptable overhead.

1 Introduction

Enterprise networks' first line of defense typically consists of firewalls and virtual private network (VPN) gateways. System administrators usually attempt to install such nodes at every point of contact between an enterprise's intranet and the public Internet. However, the security perimeter thereby achieved can be rather leaky. Users can bring to their offices mobile computers infected while used on a trip. Telecommuters can use hacked computers to connect to the enterprise's VPN. Users often install in their offices wireless access points or modems, without consulting system administrators. Users may find such rogue network nodes or mobile computers convenient, but attackers can use them to circumvent the enterprise's firewalls.

These vulnerabilities could be prevented in an enterprise network that verifies that a node's configuration conforms to the enterprise's security policies *before* accepting connection of the node to the network. This type of network policy enforcement has not traditionally been available, but currently there are several initiatives to support it, including Cisco's Network Admission Control (NAC) [1] and Microsoft's Network Access Protection (NAP) [9].

In Microsoft's NAP, before connecting to an intranet, a host sends to a network-designated server a list of the host's software components and configuration. This list indicates, e.g., what operating system and anti-virus software

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 154–165, 2005.

are installed in the host and what version, security patches, and virus definitions they have. If the server finds that the host's software is up-to-date and acceptable, the server allows the host to connect to the intranet. Otherwise, the server confines the host to a restricted network. The restricted network allows the host's software to be updated and brought into compliance with intranet policies. Cisco's NAC architecture extends this concept to control the connection not only of hosts, but also access points and other devices. Networks and nodes that support such access control functionality are expected to become common in the next several years.

However, NAC and NAP provide only weak security. Software component and configuration lists can be easily forged or modified by a malicious user. Using a forged or modified list, an attacker can circumvent NAC and NAP and gain access to a network with a node that can greatly harm the network's security.

This paper examines the question of how *secure coprocessors* could be used to harden architectures such as NAC and NAP. The Trusted Computing Group (TCG) [18] has specified secure coprocessors (called TPMs – Trusted Platform Modules) [19] that have low cost (about $4 per host) and are commercially available in an increasing number of computers by IBM, HP, and other manufacturers [14]. TPMs enable security primitives that were previously unavailable, in particular *attestation*. Attestation reveals to another party the software configuration running on a host, in a securely verifiable manner. Unlike NAC's and NAP's software component and configuration lists, attestations cannot be easily forged or modified by attackers. Section 2 describes attestation in greater detail.

We identify three challenges for building systems that use attestation, and contribute novel solutions for them. First, attestation has to be integrated with network protocols in a manner that does not defeat security. Attestation as specified by TCG is vulnerable to man-in-the-middle (MITM) attacks. We show in Section 3.1 that this vulnerability is not eliminated simply by tunneling attestation packets, e.g. using TLS [2]. We contribute a novel form of attestation, *Bound Keyed Attestation* (BKA), that can be integrated with other protocols easily and without MITM vulnerability.

Second, the operating system (OS) has to be modified such that it properly records and reports in attestations any OS modifications since the system has booted. TCG specifies TPM's hardware, boot sequence, and interfaces, but not the inner working of systems that use those interfaces, especially after boot time. Most OSs define privileged users (e.g., root) with authority to modify the OS or its configuration at any time. However, the charter of TCG's relevant working group explicitly excludes what the OS should do to maintain attestation consistency after boot time [20]. In Sections 4.1 and 4.2, we propose *TCB prelogging* and *security association root tripping* for guaranteeing such consistency.

Third, use of a TPM must not harm host safety. In addition to attestation, TPMs provide another security primitive, *sealing*. Data sealed to a given host software and configuration can be accessed only when the host software and configuration are the same. Sealing can be used for *digital rights management*. Before sending content to a receiver, a sender obtains the receiver's attestation.

The sender sends the content only if the sender deems the receiver's software as trustworthy. The receiver's software then seals the content so that it cannot be accessed by untrusted software. However, sealing can be used also for *software lock-in* [4]. An application, e.g. an editor, may seal to itself content created by the host owner. This can make it impossible for the host owner to switch to a different application or to access the data using a future host. In this sense, sealing can make a host unsafe. In Section 4.3, we propose *sealing-free attestation confinement* for securing network access without compromising host safety.

Experimental results in Section 5 show that our proposed mechanisms are effective and have acceptable overhead. We discuss related work in Section 6, and conclude in Section 7.

2 TPM Functions

To secure access to enterprise networks, we use two main TPM functions: *authenticated boot* and *attestation*. We describe these functions in this section. Readers who are already familiar with TCG secure coprocessors may skip this section.

Authenticated boot presupposes that when a host is reset, control of the CPU is transferred to a small, trusted, immutable software component. In a personal computer, this component is the BIOS boot block. The OS's boot sequence is modified such that, before each software component A passes control to another software component B that has not yet been measured, A measures B's digest, appends the digest to a *measurement log*, and compresses the digest into the TPM. A obtains B's digest using SHA-1 (Secure Hash Algorithm) [11]. SHA-1 digests are 20 bytes long, regardless of data length. This algorithm has properties such that it is infeasible to modify B without also modifying its digest, or to find a B whose digest is an arbitrary value. The TPM compresses a digest into one of its registers by concatenating the register's value and the digest, computing SHA-1 on this concatenation, and storing the resulting value back into the register. The TPM's registers are initialized to zero on host reset. They can be read, but cannot be otherwise modified. TPM register values can be used to authenticate the measurement log, whose plaintext is stored in main memory.

Attestation is a protocol that enables a remote party R to obtain and authenticate a host P's measurement log. R sends to P a nonce, i.e., a cryptographically random number that is never reused. P asks its TPM to sign a so-called *quote*, containing the nonce and current values of the TPM's registers. Presence of the nonce in the quote guarantees that a quote cannot be later replayed. The signature uses an attestation identity key (AIK). P sends the corresponding AIK certificate to R together with the quote and measurement log. TPMs generate private and public AIK pairs internally. TPMs use private AIKs only to sign quotes (as defined above), and never reveal such keys externally. The host owner obtains the AIK certificate from a so-called *privacy certifying authority*, which verifies that the host contains a properly attached TCG-compliant TPM. R authenticates the AIK certificate using the certifying authority's public key, which R is assumed to know out-of-band. R then uses the nonce and public AIK to authenticate the quote and uses the quote to authenticate the measurement log. The authenticated log reveals securely to R what software booted in P.

3 Network Protocol Enhancements

This section describes protocol enhancements for securing network access.

IEEE 802.1x [7] is a standard, widely supported protocol that can be configured to require mutual authentication between a node and a local area network (LAN) before the node is allowed to communicate through the network (beyond authentication). The participants to this protocol are the *supplicant*, i.e. the node that requests access, the *authenticator*, e.g. a switch or access point that mediates the supplicant's access to the network, and the *authentication server*, e.g. a RADIUS server that authenticates and authorizes the supplicant's access. A variety of authentication protocols can be used over 802.1x, including PEAPv2 [13]. PEAPv2 begins by creating a TLS tunnel between authentication server and supplicant, with certificate-based authentication of the former by the latter. The server then typically uses MS-CHAPv2 for password-based supplicant authentication. Finally, PEAPv2 binds the TLS and MS-CHAPv2 keys to guarantee that the respective endpoints are the same. 802.1x also enables the creation of a security association between authenticator and supplicant, with cryptographic keys for encrypting and authenticating packets sent between them. On Ethernet, such packet-level cryptography is being standardized by IEEE 802.1ae; on Wi-Fi, packet-level cryptography is enabled by IEEE 802.11i.

PEAPv2 authenticates only the supplicant's user. We propose to combine PEAPv2 with Bound Keyed Attestation (BKA). This combination enables the server to authenticate both the supplicant's host configuration and user before the server authorizes the supplicant's communication. If the server does not trust the supplicant's configuration, the server can deny access to the supplicant or confine the supplicant to a restricted virtual LAN (VLAN, as indicated, e.g., by a RADIUS *access accept* attribute).

3.1 Bound Keyed Attestation

This subsection shows that attestation, as defined by TCG, is vulnerable to MITM attacks. It then proposes BKA, a novel form of attestation that resists MITM attacks.

As shown in Fig. 1, a malicious user can use two computers, one of which is conformant to network security policies, to get past TCG-defined attestation. The user hacks the intermediate computer such that it passes to the user's conformant computer any attestation requests, and passes the corresponding attestation replies back to the authentication server. The latter cannot discern the presence of a MITM, and therefore authorizes access, even though the MITM may have a configuration in complete violation of the network's security policies.

This vulnerability is *not* eliminated by securely tunneling attestation messages, e.g. using TLS. A secure tunnel can protect the confidentiality and integrity of the messages against third-party attacks. However, if the client is not

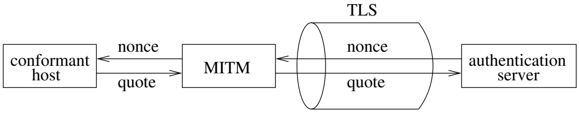


Fig. 1. TCG-specified attestation is vulnerable to MITM attacks. If a malicious user has two computers, one of which is conformant to the server's policies, the user can employ the conformant computer to get a hacked computer accepted by the server. It makes no difference if TLS is used

trustworthy, the tunnel cannot prevent the latter from acting as a MITM that relays attestation or other messages between the tunnel and another computer.

To thwart this type of attack, we propose a novel form of attestation, Bound Keyed Attestation (BKA). It differs from TCG-specified attestation in two important ways. First, it derives new keys shared by the attestation endpoints. Second, it securely binds these keys to a tunnel's keys. This binding guarantees that tunnel and attestation endpoints are the same. Consequently, a MITM attack is not possible.

All BKA messages can be transmitted using a previously established secure tunnel T (e.g., PEAPv2's TLS tunnel) with which the attestation will be bound. It is assumed that T is cryptographically secured with a dynamically generated secret key K_T that is not revealed to users. T can have states *attested* or *not-attested* (default). System policies can prevent communication other than for attestation while the tunnel is not-attested. Like a Diffie-Hellman key exchange [3], BKA uses two publicly known numbers: a prime number q and an integer α that is a primitive root of q. These numbers do not need to change. As shown in Fig. 2, the *attestation initiator* A picks two random numbers: a nonce N_A, which is never reused, and another integer $0 \leq X_A < q$, which may be reused. The initiator computes its public key $Y_A = \alpha^{X_A}$ mod q. The initiator then sends to the *attestation responder* B a *BKA request* message M_i containing q, α, Y_A, and N_A.

The responder then picks a random integer $0 \leq X_B < q$, which may be reused, and computes its public key $Y_B = \alpha^{X_B} \bmod q$. The responder computes the *attestation shared secret* as $K_{AB} = Y_A^{X_B} \bmod q$, and its *attestation binding* $B_B = \text{SHA1}(N_A|K_T|K_{AB}|Y_A|\text{“BKA response”})$, where | denotes concatenation. The responder also computes the *initiator's attestation nonce* $n_A = \text{SHA1}(N_A|K_{AB})$. The responder then gets from its TPM its quote Q_B containing n_A. Finally, the responder sends to the initiator a *BKA response* message M_r containing Y_B, a fresh nonce N_B, B_B, Q_B, and B's measurement log L_B and AIK certificate C_B. Alternatively, if the responder also wishes to obtain the initiator's attestation, the responder sends to the initiator a *BKA response-request* message M_{rr} containing the same fields as M_r.

The initiator processes M_r as follows. First, the initiator computes the attestation shared secret as $K_{AB} = Y_B^{X_A}$ mod q. The initiator then computes the expected value of B_B using K_{AB}. If the received B_B does not match this expected value, the initiator sends a *BKA error* message to the responder and returns failure (possibly the tunnel and attestation endpoints are not the same,

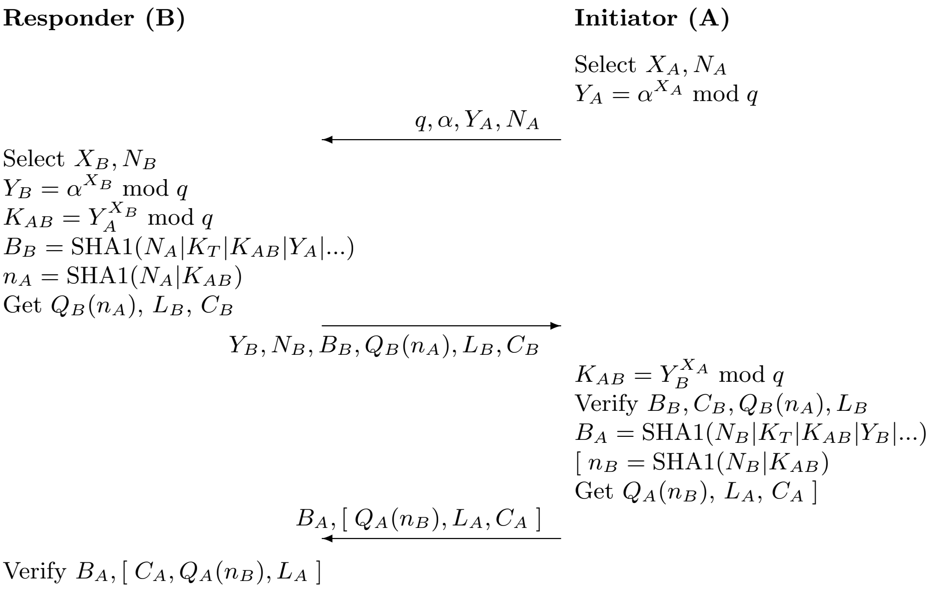

Responder (B)
Initiator (A)
Select X_A, N_A
$Y_A = \alpha^{X_A} \bmod q$
q, α, Y_A, N_A
Select X_B, N_B
$Y_B = \alpha^{X_B} \bmod q$
$K_{AB} = Y_A^{X_B} \bmod q$
$B_B = \text{SHA1}(N_A|K_T|K_{AB}|Y_A|...)$
$n_A = \text{SHA1}(N_A|K_{AB})$
Get $Q_B(n_A)$, L_B, C_B
$Y_B, N_B, B_B, Q_B(n_A), L_B, C_B$
$K_{AB} = Y_B^{X_A} \bmod q$
Verify $B_B, C_B, Q_B(n_A), L_B$
$B_A = \text{SHA1}(N_B|K_T|K_{AB}|Y_B|...)$
[$n_B = \text{SHA1}(N_B|K_{AB})$
Get $Q_A(n_B)$, L_A, C_A]
B_A, [$Q_A(n_B), L_A, C_A$]
Verify B_A, [$C_A, Q_A(n_B), L_A$]

Fig. 2. Bound Keyed Attestation (BKA) thwarts MITM attacks by guaranteeing that the endpoints of the communication tunnel (e.g., TLS) and attestation are the same. Items between brackets are needed only for mutual attestation

and a MITM attack is happening). Otherwise, the initiator authenticates C_B using the respective certifying authority's public key (known securely out-of-band), authenticates the responder's quote Q_B using n_A and C_B, and authenticates the responder's measurement log using Q_B. If any of the authentications fails, or the initiator does not recognize a measurement in the responder's log L_B as that of a trusted software component, the initiator sends a BKA error message to the responder and returns failure. Otherwise, the initiator computes its binding $B_A = \text{SHA1}(N_B|K_T|K_{AB}|Y_B|\text{“BKA success”})$. The initiator then sends to the responder a *BKA success* message M_s containing B_A, transitions T's state to *attested*, and returns success.

Processing of M_{rr} by the initiator is similar to that of M_r, except that (1) the initiator also computes the *responder's attestation nonce* $n_B = \text{SHA1}(N_B|K_{AB})$ and gets from the initiator's TPM its quote Q_A containing n_B, and (2) instead of M_s, the initiator sends to the responder a *BKA success-response* message M_{sr} containing B_A, Q_A, and A's measurement log L_A and AIK certificate C_A.

The responder processes M_s by computing the expected value of B_A and verifying that the received B_A matches it. If so, the responder transitions T's state to *attested*. Otherwise, T's state remains *not-attested*. Processing of M_{sr} is similar, except that, if the received B_A matches its expected value, (1) the responder also authenticates C_A, Q_A, and L_A, and (2) if all authentications succeed, and the responder identifies each measurement in the initiator's log as that of a trusted software component, then the responder transitions T's state to *attested*; otherwise, T's state remains *not-attested*.

Nonces N_A and N_B guarantee quote freshness and allow bound keyed attestations to be obtained frequently, without burdening the processors each time with the expensive computation of Y_A, Y_B, and K_{AB}. Performance can be improved also by caching the authentication of C_A and C_B and the values of common intermediate expressions that change infrequently. The attestation nonces n_x are one-way functions of not only the nonces N_x but also the attestation shared key K_{AB}. Therefore, the quotes are bound to the attestation shared key, which in turn is bound to the tunnel key K_T by B_x, where $x \in \{A, B\}$.

4 Operating System Enhancements

As explained in the previous section, BKA requires hosts to have TPMs. However, use of TPMs poses several OS challenges. We propose in the following subsections three novel solutions to these challenges: *TCB prelogging*, *security association root tripping*, and *sealing-free attestation confinement*.

4.1 TCB Prelogging

TCG documents specify that a system's BIOS, master boot record, boot loader, and OS kernel should be measured and their digests included in the system's measurement log. However, not only these components, but any member of the system's *Trusted Computing Base* (TCB) needs to be measured. A system's TCB is defined as the set of components whose malfunction (due, e.g., to a bug or attack) would allow the system's policies to be compromised. Therefore, the TCB includes not only the TCG-mentioned components, but also their configuration files and any privileged applications that could modify them, e.g. root-owned scripts or daemons and setuid applications. On the other hand, unprivileged user applications that are not part of the TCB can be created, configured, modified, or destroyed without compromising the system's ability to enforce policies.

We propose that each system maintain a configuration file listing the system's TCB components and respective digests. A bug in this TCB list (e.g., absence of a TCB component) could enable system policies to be violated. Therefore, the TCB list needs to include an entry for itself. By convention, the digest of the TCB list is calculated by making this entry's digest equal to zero, and the result is stored in this entry. At boot time, after the kernel mounts the file systems, the kernel appends the TCB list to the measurement log and compresses the list's digests into the TPM. Thereafter, whenever a file that is a TCB component, root-owned script or daemon or setuid application is opened or exec'ed, the kernel measures the file's digest. If this digest is different from the one last logged for the file, the kernel appends the new measurement to the log and compresses the new digest into the TPM. Thus, an authenticated measurement log will reveal whether a tampered or untrusted TCB component has run in the system since the system booted.

4.2 Security Association Root Tripping

Unprivileged users cannot cause the OS kernel or daemons to compromise the system's policy enforcement. (If that is not true, the system has a bug that needs to be fixed.) However, privileged users (e.g., root) *can* easily violate the system's policy enforcement, e.g. by using commands such as sysctl or ifconfig or by using a debugger to attach and modify privileged processes, after boot time. It can be difficult or impossible to guarantee that all such configuration modifications are captured in the system's measurement log. Moreover, TCG does not specify what to do when such modifications occur.

We propose to modify the OS such that it detects and takes appropriate action when a privileged user attempts to gain interactive access to the system (e.g. by logging in or using the su command). If there are any attestation-based security associations, the system warns the user that, if the user wants to continue, the system will immediately drop those security associations. Thus, in the case of secure intranet access, the system will destroy the keys used for packet-level cryptography, making access impossible. Furthermore, if the user wants to continue, the system appends this event to the measurement log with a well-known digest and compresses the latter into the TPM. Therefore, subsequent attestations will show that a privileged user has logged in interactively. System administrators can configure authentication servers to deny access to such supplicants, who will need to reboot before again gaining access to the network (reboot erases the measurement log, as well as any non-persistent configuration changes; persistent changes are captured by the new measurement log after reboot). These mechanisms do not preclude remote system administration or help, as long as these are performed using daemons that the network's authentication server is configured to trust.

4.3 Sealing-Free Attestation Confinement

TPMs include a function, sealing, that can be abused by applications. As discussed in the Introduction, sealing enables software lock-in, making hosts unsafe.

To enable BKA without compromising safety, we propose that the OS (1) support authenticated boot and attestation, but not sealing, and (2) allow attestation only in conjunction with network access control protocols (such as 802.1x and IPsec's IKE). Abusive applications then cannot encrypt and bind to themselves file contents, as necessary for software lock-in. When the host is disconnected, the contents would need to be sealed, but the OS does not provide this service. On the other hand, when the host is Internet-connected, abusive applications, instead of sealing, could attempt to store file encryption keys in a remote server that reveals the keys only to abusive applications in attested clients. However, the OS allows attestation only in conjunction with protocols that typically cannot go through enterprise firewalls. Because the OS confines attestation to the intranet, it thwarts software lock-in also in this case.

Table 1. Breakdown of authentication and authorization latency and projected throughput (standard deviations represented between brackets: [σ])

Step	PEAPv2	PEAPv2 + LOG	PEAPv2 + BKA
TLS	39.6 ms [0.2]	39.9 ms [0.8]	38.9 ms [0.9]
LOG		24.4 ms [0.2]	
BKA			2758 ms [263]
MS-CHAPv2	20.6 ms [0.5]	17.4 ms [0.2]	17.9 ms [0.2]
Binding TLS/MS-CHAPv2	7.0 ms [0.2]	6.8 ms [0.2]	6.8 ms [0.1]
Total	67.2 ms [0.5]	88.4 ms [0.5]	2822 ms [263]
CPU busy	22.6 ms [0.2]	23.9 ms [0.2]	116 ms [10]
Projected throughput	2650 supp/min	2510 supp/min	519 supp/min

5 Experimental Evaluation

This section evaluates the performance impact of the proposed mechanisms. Reported results are averages of six measurements.

We implemented the proposed OS enhancements on FreeBSD 4.8 and installed this OS on an IBM ThinkPad T30 computer with 1.8 GHz Pentium 4 CPU, 256 MB RAM, TPM version 1.1b, TPM-aware BIOS, and built-in 802.11b interface. The master boot record and GRUB were modified for measuring digests and compressing them into the TPM as specified by TCG. We measured a total boot time of 20.08 s (σ = 0.12) before and 20.15 s (σ = 0.17) after our modifications. Although TCB prelogging and file digest measuring impose overheads, they are completely dominated by other boot costs. During system operation, each file digest measurement can be cached, and need not be repeated while the file is not modified [15]. Because TCB components change infrequently, file digest measurements can be expected to have little impact on steady-state performance. Security association root tripping affects only certain commands (e.g., login and su) and only when used by privileged users. Therefore, it also has negligible performance impact, as does sealing-free attestation confinement.

We integrated PEAPv2/802.1x with BKA on the FreeRADIUS authentication server [5] and Open1x supplicant [12]. In order to estimate a comparison with NAP, we alternatively integrated PEAPv2/802.1x with a NAP-like LOG protocol. Both in BKA and in LOG, the supplicant sends its list of software components and configuration to the authentication server, who may approve it or not. However, unlike BKA's list, LOG's list can be forged by the supplicant. We installed FreeRADIUS on a Dell Dimension 4550 computer with 2.4 GHz Pentium 4 CPU, 256 MB RAM, and unmodified FreeBSD 4.10. We installed Open1x on the aforementioned IBM T30 computer. As authenticator, we used a Cisco Aironet 1100 802.11b access point connected to the authentication server via Fast Ethernet. We used BKA with precomputed modulus (1024-bit) and primitive root. Table 1 shows that the time it takes for the supplicant to connect to the network increased from about 67 ms to 88 ms with LOG or 2.8 s with BKA. To understand the source of BKA's large overhead, we measured the time it takes for the supplicant to obtain a quote from its TPM, and found it to be 2.5 s ($\sigma = 0.2$). Therefore, most of the latency added by BKA is due to the TPM. However, even with a low-cost, slow TPM, such as the one we used, the connection latency is acceptable.

In order to evaluate the impact of LOG or BKA on the CPU utilization of the authentication server, we instrumented the OS's idle loop and interrupt vector. The instrumented OS uses the CPU's built-in cycle counter to measure CPU idle time, excluding interrupts. We also instrumented FreeRADIUS to measure the total time necessary to authenticate and authorize supplicant access. Table 1 shows how long the authentication server's CPU was busy while processing a single supplicant, and the projected throughput (load for saturating the CPU). LOG and BKA reduced projected throughput from roughly 2650 to 2510 or 519 supplicants/minute, respectively. BKA's large overhead is due to its use of public-key algorithms for key exchange and authentication of certificates and quotes, as well as repeated applications of SHA-1. Although BKA's overhead is substantial, the projected throughput may be acceptable for medium-size networks. Note that, in our scheme, a supplicant imposes no load on the authentication server after the latter authorizes access. In larger networks, load can be distributed simply by using multiple low-cost authentication servers, such as the one we used.

6 Related Work

NAC [1] and NAP [9] are the architectures most closely related to our work. Because NAP uses DHCP for controlling access to enterprise LANs, malicious users can circumvent it simply by using a static networking configuration. We use 802.1x instead of DHCP to avoid this security loophole. NAC is currently implemented on certain routers, using a proprietary access control protocol. Unlike our solution, such an implementation does not prevent unauthorized access to LANs that a malicious user might directly connect to.

Our TCB lists are similar to what Tripwire [17] uses to verify host integrity. However, unlike Tripwire, we enable administrators to verify host integrity remotely. Bear [8] is a Linux-based OS that supports TPMs and uses similar lists, signed by trusted system administrators. Bear's Enforcer verifies at load time whether a file's digest matches its value on the signed list. In case of a mismatch, Enforcer takes the action specified on the list (e.g., return failure). Bear does not support attestation, relying instead on certificates with special semantics, and has no protection against privileged users. Security association root tripping (Section 4.2) would significantly improve Bear's security.

TcgLinux [15] is another OS with TPM support. Because tcgLinux does not have a TCB list, it logs and compresses into the TPM digests of *all* files that are executed, and requires shells and other programs to be modified to do the same with their security-sensitive scripts and configuration files. This design makes it harder to verify that all TCB configuration files are being measured. It also unnecessarily exposes in attestations the execution of unprivileged programs, reducing user privacy. TcgLinux has mechanisms to prevent privileged users from making system modifications that might not be detected by attestation. It does allow, however, any modifications that would be detected by attestation. TcgLinux attestations have been used to secure VPN access [16]. However, that solution has several shortcomings. First, TCG-specified attestation is vulnerable to MITM attacks, as shown in Section 3.1. Second, tcgLinux would be vulnerable to MITM attacks even if BKA were used, because it does not prevent privileged users from reading secret keys and other parameters of VPN tunnels. In contrast, security association root tripping would close the VPN tunnels before such reading would be possible. Third, the VPN gateway has to verify fresh attestations of each client frequently. If attestation frequency is low, users may be able to connect an insecure node to the VPN long enough to cause harm. On the other hand, our experimental results suggest that high attestation frequencies could severely limit the throughput of the VPN gateway. In contrast, security association root tripping achieves security with a single attestation at the beginning of each client's session.

Microsoft's NGSCB architecture [10] uses TPMs and divides the system into trusted and untrusted halves. The untrusted half runs a conventional OS, including file system and network protocol stack. On the other hand, the trusted half secures user credentials and keys for digital rights management. It uses storage and communication services provided by the untrusted half. NGSCB requires special CPUs with Intel's LaGrande Technology (LT). Terra [6] is an OS with similar architecture, but uses a virtual machine monitor instead of LT. It is unclear how NGSCB or Terra would be used for securing network access, because those systems cannot guarantee the integrity of the untrusted half's configuration.

7 Conclusions

Firewall-based network security perimeters can be leaky. Architectures such as NAC and NAP attempt to plug perimeter leaks by preventing nodes that do not conform to a network's security policies from joining the network. However, those architectures are vulnerable to malicious users. TCG has specified inexpensive secure coprocessors that could harden those architectures. However, several challenges need to be overcome, including how to prevent MITM attacks, undetected changes to system files, tampering by privileged users, and software lock-in. We proposed novel solutions to these challenges: bound keyed attestation, TCB prelogging, security association root tripping, and sealing-free attestation confinement. We integrated these mechanisms with FreeBSD and PEAP over 802.1x. Experimental results show that our techniques allow secure coprocessors to protect access to enterprise networks robustly, safely, and with acceptable overhead.

References

1. Cisco: Network Admission Control. [Online] http://www.cisco.com/en/US/netsol/ns466/networking_solutions_sub_solution_home.html

2. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. IETF, RFC 2246, Jan. 1999. [Online] `ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt`
3. Diffie, W., Hellman, M.: New Directions in Cryptography. In *Transactions on Information Theory*, IEEE, 1976, 22:644-654.
4. Felten, E.: Understanding Trusted Computing. In *Security and Privacy*, IEEE, May/June 2003, pp. 60-62. [Online] `http://www.princeton.edu/~echi/ele572/Felten\%20-\%20Understanding\%20trusted\%20computing.pdf`
5. FreeRADIUS: Homepage. [Online] `http://www.freeradius.org/`
6. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A Virtual Machine-Based Platform for Trusted Computing. In *Proc. 19th Symposium on Operating System Principles*, ACM, 2003. [Online] `http://www.stanford.edu/~talg/papers/SOSP03/terra.pdf`
7. IEEE: Port-Based Network Access Control. 802.1x Std. (2001) [Online] `http://standards.ieee.org/getieee802/download/802.1X-2001.pdf`
8. Marchesini, J., Smith, S., Wild, O., Stabiner, J., Barsamian, A.: Open-Source Applications of TCPA Hardware. In *Proc. 20th Annual Computer Security Applications Conference*, ACSAC, Dec. 2004. [Online] `http://www.cs.dartmouth.edu/~carlo/research/bearapps/bearapps.pdf`
9. Microsoft: Network Access Protection. [Online] `http://www.microsoft.com/windowsserver2003/technologies/networking/nap/default.mspx`
10. Microsoft: Next Generation Secure Computing Base – Technical FAQ. July 2003. [Online] `http://www.microsoft.com/technet/security/news/ngscb.mspx`
11. NIST: Secure Hash Standard. Federal Information Processing Standards Pub. 180-1, Apr. 1995. [Online] `http://www.itl.nist.gov/fipspubs/fip180-1.htm`
12. Open1x: Homepage. [Online] `http://www.open1x.org/`
13. Palekar, A., Simon, D., Salowey, J., Zhou, H., Zorn, G., Josefsson, S.: Protected EAP Protocol (PEAP) Version 2. IETF. Internet Draft, Oct. 2004. [Online] `ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-josefsson-pppext-eap-tls-eap-10.txt`
14. Pearson, S. (ed.): Trusted Computing Platforms – TCPA Technology in Context. Prentice Hall, 2003.
15. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proc. Security Symposium*, USENIX, Aug. 2004. [Online] `http://www.usenix.org/publications/library/proceedings/sec04/tech/sailer.html`
16. Sailer, R., Jaeger, T., Zhang, X., van Doorn, L.: Attestation-based Policy Enforcement for Remote Access. In *Proc. 11th Conference on Computer and Communications Security (CCS)*, ACM, Oct. 2004. [Online] `http://portal.acm.org/citation.cfm?id=1030083.1030125`
17. Tripwire.org: Homepage. [Online] `http://www.tripwire.org/`
18. Trusted Computing Group: Homepage. [Online] `https://www.trustedcomputinggroup.org/home`
19. Trusted Computing Group: Trusted Computing Platform Alliance (TCPA) Main Specification Version 1.1b. [Online] `https://www.trustedcomputinggroup.org/downloads/Main_TCG_Architecture_v1_1b.zip`
20. Trusted Computing Group: Work Group Charter Summary. 2004. [Online] `https://www.trustedcomputinggroup.org/downloads/Work_Group_Charters_Summary.pdf`

Trusted Security Devices for Bandwidth Conservation in IPSec Environments

C.D. Mano and A. Striegel[⋆]

Department of Computer Science and Engineering,
University of Notre Dame,
Notre Dame, IN 46556, USA
{cmano, striegel}@nd.edu

Abstract. Information security a is constant concern of Internet data. One security solution is IPSec, which is a set of protocols that provides both data confidentiality and authenticity. Another concern is the last mile bandwidth limitation on many Internet connections. This problem can be mitigated by bandwidth conservation techniques such as Application Layer and Stealth Multicast (SMC). Combining IPSec and multicast techniques would be ideal, but is not possible due to the nature of encrypted data and the requirements of multicast messages. We present the concept of a Trusted Security Device (TSD) which provides efficient bandwidth usage while maintaining security levels offered by IPSec. A TSD cooperates with clients and servers while implementing SMC technology. Minor modifications to clients and servers are necessary to enable discovery, key exchange, and communication between clients, servers, and TSDs. TSD technology is applicable to streaming data where confidentiality, authentication, and bandwidth conservation are concerns.

1 Introduction

Information security is a constant concern of data transmitted through an electronic network. Security can be increased by using various data encryption techniques. Encryption techniques can be classified into two basic categories, asymmetric and symmetric. Asymmetric, or public key, encryption can be used to authenticate senders of data, while symmetric keys are typically used to provide confidentiality.

Secure Socket Layers (SSL) [1] and the IP Security Protocol Suite (IPSec) [2] are two popular methods of creating secure communication channels between two parties on the Internet. SSL is an application based technique that is commonly used in web-based communication such as web browsers. IPSec is implemented at the network layer and therefore does not rely on the support of higher level applications. Both technologies use a combination of asymmetric and symmetric encryption to provide both authentication and confidentiality to network data.

[⋆] This research was supported in part by the National Science Foundation through the grant CNS03-47392.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 166–177, 2005.

It is becoming increasingly common for home and small business Internet users to serve data to remote clients. Streaming multimedia such as audio and video, as well as online game hosting are a few examples of content that is being served by these users. Subscription based multimedia streams require a technology such as SSL to prevent theft of service. IPSec can be used in gaming environments, as in Microsoft's XBox Live, to prevent players from cheating by receiving extra information or transmitting false information.

A typical home or small business office has a limited bandwidth connection to the Internet. In cases such as Asymmetric DSL, the upstream bandwidth is even more constrained than the downstream. Bandwidth conservation techniques such as Application Layer Multicast (ALM) and Stealth Multicast (SMC)[3] are able to reduce these bandwidth demands, but are not compatible with security protocols such as those used in SSL or IPSec. This leaves hosts to choose between security and bandwidth while the situation demands that both exist together.

The marriage of bandwidth conservation and information security creates new and interesting problems which must be solved. In [4], the authors analyze security issues as they apply to multicast. In particular the topics of authentication, confidentiality, and key management are discussed. In [5] and [6] efficient key management techniques are presented. While these studies are effective in a traditional multicast infrastructure they do not apply to ALM or SMC environments.

We present the concept of a Trusted Security Device (TSD) as a solution to this problem. A TSD works in conjunction with clients and servers to utilize bandwidth conservation methods while maintaining secure communications. Section 2 discusses the motivation for this work as well as background information about IPSec and SMC. Section 3 presents the concept of the TSD and discusses communication and encryption protocols which are needed for the unit to function properly. An online gaming scenario is used to explain the entire process. Section 4 briefly discusses other applications of TSD technology, and 5 presents its possible weaknesses. The final section is a summary of this work.

2 Motivation and Background

Consider a case where a server is utilizing SSL to send a subscriber-based live music stream. The purpose of SSL is to restrict the audience to those who have paid the appropriate fees. SSL is a point-to-point protocol, so a distinct unicast stream is needed to broadcast to each subscriber. The protocol could be adjusted so that all clients, following authentication, would receive the same symmetric key so encrypted data would be identical, enabling SMC to provide bandwidth conservation services, as will be explained later in this section. However, while eavesdropping would still be prevented, the source of the data cannot be authenticated, which may pose a problem.

This problem is evident in the case where the data is a stream of stock prices. If a group key is used for the encryption we know that the data is coming from a member of the group, but we cannot authenticate which member of the group is sending it. This is a risky proposition when your hard earned money is at stake! The remainder of this section discusses IPSec and SMC, and provides insight into reasons for their inability to work together in the streaming data scenarios presented.

2.1 IPSec

IPSec is a set of protocols designed to provide security services at the IP Layer. With data encryption being handled at this layer it provides security to networked applications that do not have encryption capabilities such as SSL built in to them. Virtual Private Networks (VPN) are a popular implementation of IPSec which is commonly used to create secure connections between a user and an entity such as a company. A telecommuting employee may use a VPN to log into their company's network so all data that traverses the Internet is secure.

IPSec offers two protocols for handling network traffic security, IP Authentication Header (AH) [7] and Encapsulating Security Payload (ESP) [8]. The protocols are similar in that they are able to provide authentication, integrity, and replay protection. AH also provides integrity of the IP header and non-repudiation. ESP provides confidentiality via encryption. The two protocols can be used together to obtain all required security characteristics. In this study we are particularly concerned with the following security characteristics:

- Confidentiality
- Integrity
- Source Authentication
- Replay Protection

As mentioned previously, in order for SMC to work, a group symmetric key, rather than pairwise unique symmetric keys, must be used, but this eliminates the source authentication characteristic which is required. Another option is the use of a ticket, which is a smaller piece of data, such as a checksum of the message, encrypted with a private key and added to the message. The receiver could verify the ticket which provides source authentication. The computational requirements of asymmetric key encryption is still a problem however, as this must be done for every packet being sent.

2.2 Stealth Multicast

SMC is a method of converting redundant unicast Internet traffic into efficient multicast packets. This is enabled by the use of a module called a Virtual Group Detection Manager (VGDM) [3]. A VGDM performs similarity analysis on packet payloads and, based on predefined heuristics, caches the packet or forwards it on to its destination. A cached packet will become a member of a virtual group, or will simply be forwarded on as normal. Virtual groups are comprised of a single packet payload along with the destinations of all packets with the identical payload. A group is used to create a single multicast packet to be delivered via SMC. Multicast packets are received by an SMC enabled device on the client side where the packet is converted back to the original unicast packet.

Again, this method is dependent on the ability of the VGDM to identify similar packet payloads in order to create virtual groups. Encryption of data creates a problem because unless the same key is used for all transmissions identical plaintext creates different ciphertext. As was discussed before, a group symmetric key allows the similarity detection to take place, but we suffer because source authentication is not possible.

3 Trusted Security Device

We propose the concept of a Trusted Security Device to solve this problem. A TSD works in conjunction with clients and servers to provide the data security characteristics of confidentiality, integrity, source authentication, and replay protection. It is even effective in a heterogeneous environment where not all clients are enabled with a TSD unit. The TSD works as a trusted party which is delegated the responsibility of providing security. Minor modifications are required of clients and servers in order to facilitate direct communication with the TSD. VGDM capabilities from SMC are incorporated in order to reduce the amount of redundant traffic sent to the Internet.

We illustrate the functionality and use of a TSD with a possible real life implementation of an XBox Live gaming system. XBox Live is a service which allows XBox owners to host networked games across the Internet. It is one of many scenarios where home users may host servers and distribute streaming or other content through the Internet.

3.1 Architecture

Figure 1a is a typical XBox Live architecture consisting of multiple XBox console machines, with one being designated as the game server (XGS), and the XBox Live Server (XLS). Using a custom version of IPSec, Microsoft's XBox communicates and authenticates with the XLS. Authentication is made possible by a 2048-bit asymmetric key which is hardwired into the XBox. Once an XGS is authenticated it is listed as an open server for other players to join. An XBox client (XC), authenticated in the same manner, is able to join a game on a listed game server. Following the coordination of the game server and clients, communication travels directly between the XBox consoles. While Microsoft does not

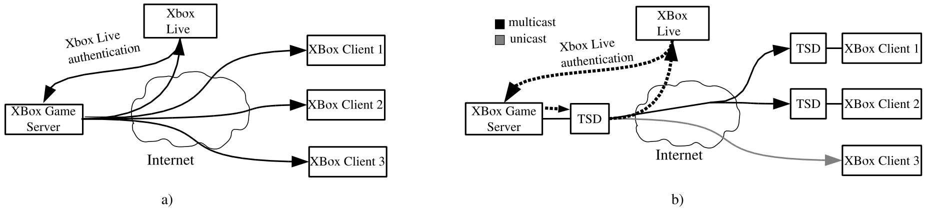


Fig. 1. Configuration and data flow of typical XBox Live setup

release details of their protocols or implementation, our analysis of XBox Live traffic shows that communication is done with encrypted unicast UDP packets. We will assume that each connection between a client and game server is established as a secure IPSec connection. This results in encryption using a different shared secret key between the server and each client. It is important to have this secure communication for game data to prevent cheating. In an unsecured environment unscrupulous players may collect information about the game or transmit false messages to other players in order to gain an unfair advantage.

Figure 1b shows a similar environment as previously described with the addition of TSDs on the game server and two of the three client machines. The TSD lies in-line between the XBox and the router. While the TSD will analyze and retransmit much of the XBox game data, for all other communication, including authentication from the XBox Live server, messages from clients, and dhcp requests, the TSD acts as a pass-through device. The following subsections describe the process of TSD discovery and the encryption protocols used to provide necessary encryption while allowing a VGDM to improve network efficiency.

3.2 Discovery

Each individual XBox, including the game server, must discover whether or not a TSD exists in their local environment. Because of the high level of trust which must exist between an XBox and TSD this requires a protocol which allows an XBox to discern between an authorized TSD and a spoofed, and possibly malicious, one.

First, authentication between an XBox and the XLS is established as normal. This is enabled by the hardwired 2048-bit asymmetric key, and results in a secure IPSec connection between the XBox and the XLS. Once this connection has been established an XBox can discover its local TSD.

Similar to a Public Key Infrastructure (PKI) [9], an XBox uses the XLS as a trusted third-party to establish the authenticity of the TSD. First, the XBox must broadcast a message to its LAN.

$$XBox \rightarrow BCast: \ (M, ID, N, ttl = 1)$$

where

$$N = E(Nonce, K_s)$$

This message is comprised of four items. First, a message M states that the message is requesting discovery of a TSD. Next is an identification number of the XBox followed by a nonce encrypted with a secret symmetric key generated by the XBox. This is for the purpose of authenticating a future message from the XLS. Finally the message has a TTL of one hop, which is sufficient because of the network topology requirements of the TSD.

If any device other than a TSD receives this broadcast message it will be discarded. A TSD will use the message as a signal to send a message to the XLS in order to authenticate itself with its local XBox.

In similar fashion to the XBox, the TSD authenticates itself to the XLS via a hardwired asymmetric key.

$$TSD \rightarrow XLS : \; E(ID + N + ST_s, TSD_{priv})$$

The XLS is able to verify that the message is coming from a valid TSD because it holds the corresponding asymmetric key of the secret hardwired key TSD_{priv}. The message includes the identification number and the encrypted nonce that the TSD received from the local XBox. The identification number is the same number as is held by the XLS already, so the XLS can identify the source of the original discovery request. The key ST_s is a symmetric key generated by the TSD that is established for communication between itself and the XBox. The XLS then sends the following message to the XBox.

$$XLS \rightarrow XBox : \; E(''TSD/OK'' + N + ST_s, XS_s)$$

where XS_s is the secret symmetric key used for the IPSec connection between the XBox and the XLS. The XBox decrypts the message, authenticating the source, and obtains the message that a local TSD has been authenticated. The authentication of the TSD is strengthened by the decryption

$$XBox : \; D(N, K_s) = Nonce'$$

and a check to verify that $Nonce'$ equals $Nonce$, the original value the XBox calculated. If $Nonce$ is validated then the XBox sends a message to the XLS confirming that the TSD is valid, at which point the XLS flags the XBox as being TSD enabled.

The XBox now holds the key ST_s which will be used for secure, authenticated communication with the TSD. The location of the TSD is unknown, so the XBox broadcasts a message similar to the initial authentication request.

$$XBox \rightarrow BCast : \; E(ConnectionRequest, ST_s)$$

Only the TSD will understand the message, as it generated ST_s, so it replies, establishing a secure communication channel.

This protocol allows an XBox to identify and authenticate a TSD in its local network without prior knowledge of its existence. A spoofed TSD and message replay are the two attacks which we are concerned with here. A spoofed XBox is not a concern because we assume it is not possible based on the XBox authentication protocol. The hardwired asymmetric key in the TSD prevents spoofing in exactly the same way as the XBox. The nonce prevents replay attacks because it is unique to each authentication request and would therefore be invalid for future requests. The secret symmetric key is also unique to each request, adding additional strength against replay attacks. The only unencrypted message is the initial authentication request from the XBox. A spoofed authentication request would be defeated because an XBox cannot be spoofed, and an authorized XBox would have to verify the original encrypted nonce in the latter stages of TSD discovery.

Now each XBox knows whether or not it has a TSD and is ready to join a game. When an XC joins an XGS an IPSec connection is established just as in the standard architecture. This results in a unique symmetric key, SC_s, between each XGS-XC pair. If game play started at this point it would proceed just as in the standard configuration. The next step is the delegation of keys, which allows the TSD to take over security responsibilities.

3.3 Trust and Delegation

There are three types of key exchanges needed to create a secure environment. First is the key exchange between the XGS and its local TSD. Second is between the TSD of the XGS and the TSD enabled XCs. The final exchange is between the XCs and their corresponding local TSD. Table 1 summarizes these keys which will be used for communication during game play. In the notation: S = Server, C* = Client (* represents identification number), and T = local TSD. So T in ST refers to the server's local TSD and in C^1T represents C^1's TSD. This exchange takes place at the beginning of each game, but because players can join or leave mid-game, is can be done for a single player at any time.

First the XGS must transfer a list of XCs along with their symmetric keys SC_s^* to its local TSD.

$$XGS \rightarrow TSD_{XGS}: \; E(XC\ List + Key\ List, ST_s)$$

Next, TSD$_{XGS}$ must determine which of the XCs are TSD enabled. This is accomplished by sending each XC an inquiry using the appropriate symmetric key obtained from the XGS.

$$TSD_{XGS} \rightarrow XC_*: \; E(TSD?, SC_s^*)$$

The TSD$_{XGS}$ intercepts replies and can then divide the XCs into two group, TSD enabled and non-TSD.

TSD_{XGS} must now generate a shared group key for multicast communication as well as an asymmetric key pair to be used for authentication. This is not the same as the 2048-bit key used for authentication with the XLS, but is a smaller key used specifically for the current communication. It then distributes the keys to TSD group members. Assuming a scenario such as Figure 1b.

$$TSD_{XGS} \rightarrow C_1: \; E(SGs + S_{pub}^{tsd}, SC_s^1)$$

Table 1. Notation of encryption keys

Key	Notation
Symmetric Key for S and C*	SC_s^*
Symmetric Group Key	SG_s
Asymmetric Keys for S's TSD	S_{priv}^{tsd}, S_{pub}^{tsd}
Symmetric Key for S and local TSD	ST_s
Symmetric Key for C* and local TSD	C^*T_s

Table 2. Summary of Key Possession

Unit	Keys
XGS	SC_s^*, ST_s
TSD$_{XGS}$	SC_s^*, ST_s, SG_s, S_{priv}^{tsd}
Non-TSD C$_*$	SC_s^*
TSD Enabled C$_*$	SC_s^*, C^*T_s
TSD$_{C_*}$	SC_s^*, SG_s, C^*T_s, S_{pub}^{tsd}

$$TSD_{XGS} \rightarrow C_2 : \; E(SGs + S_{pub}^{tsd}, SC_s^2)$$

At this point TSD_{XGS} cannot communicate directly with the TSDs of the XCs. The final step enables this communication by each XC transferring the group and private keys just received, and the shared key SC^* to the local TSD.

$$XC_* \rightarrow TSD_{C_*} : \; E(SGs + S_{pub}^{tsd} + SC^*, C^*T_s)$$

Now all keys necessary for the communication protocol have been generated and shared. A summary of each unit and the keys they possess are listed in Table 2. Note that the TSD enabled C_* merely passes the keys received from the TSD_{XGS} to the local TSD and does not need to store them for future communication. The communication protocol, in conjunction with these keys, creates a secure environment similar to the basic XBox scenario, while adding the capability of a VGDM to allow a more efficient use of bandwidth. This process will be shown as a step-by-step progression in the following subsection.

3.4 Message Processing

We are only concerned with the flow of a message from an XGS to an XC. Messages in the opposite direction proceed just as before using the established IPSec channel between the game server and client as no advantage can be gained in terms of bandwidth conservation in this direction.

Figure 2 shows the flow of messages sent from an XGS to its clients. Assume the server generates a message, M, which will be sent to clients C_1, C_2, and C_3,

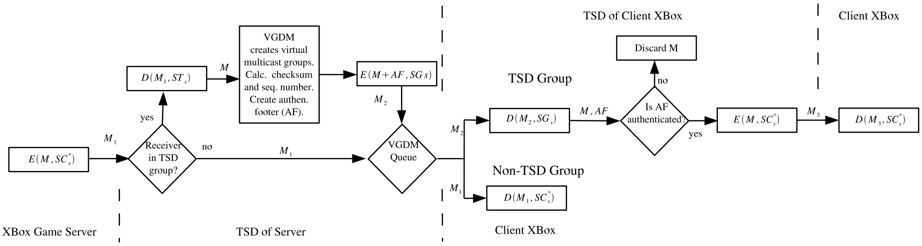


Fig. 2. Flow of messages sent from the XBox Game Server to XBox Clients with and without a TSD

where C_1 and C_2 each have a TSD (Figure 1b). The game server encrypts and sends messages as it would in a standard, non-TSD, configuration.

$$XGS \rightarrow TSD: \ E(C_1 + M, SC_s^1) = M_1^{'}$$

$$XGS \rightarrow TSD: \ E(C_2 + M, SC_s^2) = M_1^{''}$$

$$XGS \rightarrow TSD: \ E(C_3 + M, SC_s^3) = M_1^{'''}$$

The messages are intercepted by the TSD are checked to see if the destination XC is TSD enabled. TSD group messages are decrypted using the appropriate symmetric key.

$$TSD: \ D(M'{}_1, SC_s^1) = M$$

$$TSD: \ D(M''{}_1, SC_s^2) = M$$

Client C_3 is not part of the TSD group, so its message is simply placed in the VGDM queue.

$$TSD \rightarrow C_1: \ M_1^{'''}$$

Non-group messages are queued before delivering in order to minimize delivery time differences between originally temporally close messages.

The messages which are destined for TSD group members are placed in the VGDM staging area for similarity detection to take place. The VGDM is the part of SMC technology which creates multicast packets for delivery on networks lacking traditional multicast infrastructure. The VGDM analyzes data payloads and creates a message group based on the similarity of the data payloads. The result is a single message that is to be delivered to multiple XCs.

Prior to encrypting the payload of the group message an authentication footer (AF) is created (Figure 3). The AF is comprised of an MD5 hash of the message and sequence number. The AF is encrypted with the private key of the TSD, appended to the message and then the entire message is encrypted with the group key.

$$TSD \rightarrow C_1, C_2: \ E(M + E(AF, S_{priv}^{tsd}), SG_s) = M_2$$

The AF is dependent on the level of security required for a particular application. In this application a 2-byte sequence number and 4-byte hash would be sufficient.

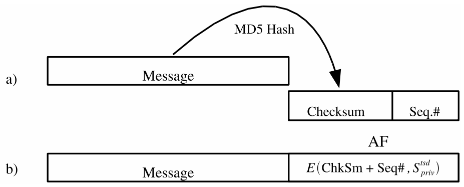


Fig. 3. An authentication footer is an encrypted 4-byte MD5 hash of the message and a 2-byte sequence number. This provides source authentication for XBox clients

The symmetric group key offers confidentiality and group authentication, but does not authenticate the sender within the group. Public key encryption of the entire message would allow for this authentication, but computational requirements for public key encryption make this unrealistic. The AF provides three important aspects of data security. First, the sequence number prevents replay attacks. More details will be given during the description of the receiver TSD. Second, the checksum verifies the integrity of the data, which ensures that the message has not been tampered with. Finally, encryption with the private key of the TSD provides for source authentication. The public key encryption here is minimal as compared to encrypting the entire message, minimizing the latency of delivering the message.

The VGDM then transmits the group messages via ALM or some other multicast method. The client side TSD receives the multicast packets and decrypts both the message and the resulting AF.

$$C_*TSD:\ D(M_2, SG_s) = M + AF$$

$$C_*TSD:\ D(AF, S^{tsd}_{pub}) = checksum + seq.number$$

The checksum is validated by calculating an MD5 hash of M and comparing the results. The sequence number is compared to the previous number and if it falls within a predetermined range, such as plus or minus 10, the message is accepted. The message is then encrypted and delivered to the XC.

$$C_*TSD \rightarrow C_*:\ E(M, SC^*_s) = M_3$$

Notice that each XBox uses the same keys for sending and receiving as they would in a non-TSD scenario. Also, the only additional information they hold concerns their own local TSD. The TSD of the XGS is the only entity that holds all knowledge of the TSD enabled XCs. So the only modification needed to the XBox is to enable it to discover a TSD and exchange keys.

In an online gaming environment backward and forward confidentiality are typically not a concern. In other environments it may be, and can be easily handled with key updates. Sophisticated key update protocols, typically for large audiences, are discussed in [4]. Generally speaking, when a client joins or leaves, or at set intervals, the new keys must be distributed to the TSD group members. In a gaming scenario, with a limited number of participants, it would probably be sufficient to perform this operation at the start of each game, but can be done as needed. The SG_s keys and the S^{tsd}_{pub} and S^{tsd}_{priv} keys would need to be regenerated by the TSD of the XGS and appropriately distributed through the secure IPSec connections just as was done during the initial key exchanges.

The security attributes of data integrity and confidentiality, replay protection, and sender authentication have all been maintained with the addition of the TSD in the online gaming framework. At the same time the TSD has enabled a technology such as SMC to efficiently transmit data via a multicast protocol. This allows hosts with constrained or limited upstream bandwidth to provide streaming multimedia content while reducing QoS problems associated with bandwidth constraints. Bandwidth conservation is obviously an advantage to the content host, but it is also positive for the receivers. This enables them to take advantage of a provider's efficient data stream, improving the QoS of the presentation on their end.

4 Other Applications

We have presented the TSD concept within the framework of a popular online gaming architecture. However, the capability of the TSD is not limited to such a scenario. The true ability of the TSD is the offering of data confidentiality and source authenticity while reducing distribution bandwidth requirements.

The target application for this technology is anything that needs to provide information security while delivering content to multiple receivers. Previously we mentioned the streaming stock price example. This, as well as most information data streams, is an ideal place for TSD technology. General public subscription based audio and video streams may or may not be candidates depending on source authentication needs. However, for private groups, such as corporate, government, or military organizations, source authentication is critical and TSD services would be a great benefit.

We have begun to investigate how the TSD would fit into a distributed backup system. This would allow secure data to be efficiently transferred to multiple backup sites simultaneously. This scenario is more complex than streaming media because of the effects of dropped packets. We will be investigating a method of efficiently incorporating the ability to handle this in the future.

5 Weaknesses

The addition of a TSD obviously offers a new point of attack for malicious users. However, the authentication process for the TSD is similar to that of an XBox, so in terms of security strength they can be considered equal. In other scenarios the TSD would still be authenticated in a way that provides equal security strength to the existing infrastructure.

A system that does not require modification of existing systems would be ideal in terms of ease of deployment and scope of use. The TSD, however, does require modifications to be made in order for trust to be established and keys to be exchanged. This requirement, while necessary and unavoidable, prevents the TSD from becoming a general purpose network appliance, but does position it as a revenue generating add-on product.

6 Summary

The marriage of bandwidth conservation and information security can be performed by the implementation of a trusted security device. Bandwidth conservation techniques such as ALM or SMC can be used to reduce redundant network traffic and do not suffer the deployment problems of traditional multicast. SMC relies on the ability to detect identical data payloads to create virtual multicast groups for efficient delivery of data. IPSec is a protocol for secure data transfer, but does not work with a SMC because SMC is not able to detect similar data when encrypted with different keys.

The TSD is a trusted device to which clients and servers delegate the responsibility of providing authentication and confidentiality of data. TSD implementation requires minimal modification to servers and clients to be able to function together. Discovery, key exchange, and the encryption protocol are the three major steps to creating a secure infrastructure with TSD.

A TSD infrastructure allows secure streaming data to be efficiently broadcast across the Internet. It allows bandwidth limited hosts to increase their audience while minimizing bandwidth costs. End users will benefit by receiving more efficient TSD enabled streams which increase the QoS over the traditional unicast streams from the same server.

References

1. P. K. Alan Freier, Philip Kariton, “The SSL protocol: Version 3.0,” Netscape Communications, Inc., Mountain View, CA, March 1996.
2. S. Kent and R. Atkinson, “Security Architecture for the Internet Protocol,” Internet Engineering Task Force: RFC 2401, November 1998.
3. A. Striegel, “Stealth multicast: A catalyst for multiccast deployment,” in *Proceedings of IFIP Networking*, Athens, Greece, May 2004, pp. 817–828.
4. T. Hardjono and G. Tsudik, “IP multicast security: Issues and directions,” *Annales de Telecom*, 2000.
5. I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, “Key management for secure internet multicast using boolean function minimization techniques,” in *Proceedings IEEE Infocomm’99*, vol. 2, 1999, pp. 689–698.
6. G. Ateniese, M. Steiner, and G. Tsudik, “New multiparty authentication services and key agreement protocols,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 628–639, 2000.
7. S. Kent and R. Atkinson, “IP Authentication Header,” Internet Engineering Task Force: RFC 2402, November 1998.
8. S. Kent and R. Atkinson, “IP Encapsulating Security Payload (ESP),” Internet Engineering Task Force: RFC 2406, November 1998.
9. R. Housley, W. Polk, W. Ford, and D. Solo, “Internet X.509 Public Key Infrastructure,” The Internet Society: RFC 3280, April 2002.

LIPS: Lightweight Internet Permit System for Stopping Unwanted Packets

Changho Choi[1], Yingfei Dong[2], and Zhi-Li Zhang[1,*]

[1] Dept. of Computer Science, Univ. of Minnesota, Minneapolis, MN 55455
choi,zhzhang@cs.umn.edu
[2] Dept. of Electrical Engineering, Univ. of Hawaii, 2540 Dole St., Honolulu, HI 96822
TEL:01-808-956-3448, FAX: 01-808-956-3427
yingfei@hawaii.edu

Abstract. In this paper, we propose a *Lightweight Internet Permit System (LIPS)* that provides a lightweight, scalable packet authentication mechanism for ensuring traffic-origin accountability. LIPS is a simple extension of IP, in which each packet carries an *access permit* issued by its destination host or gateway, and the destination verifies the access permit to determine if a packet is accepted or dropped. We will first present the design and the prototype implementation of LIPS on Linux 2.4 kernel. We then use analysis, simulations, and experiments to show how LIPS can effectively prevent protected critical servers and links from being flooded by unwanted packets with negligible overheads. We propose LIPS as an domain-to-domain approach to stop unwanted attacks, without requiring broad changes in backbone networks as other approaches. Therefore, LIPS is incrementally deployable in a large scale on common platforms with minor software patches.

Keywords: Network Security, IP Spoofing, Denial of Service, Unwanted Packets.

1 Introduction

One of the key security issues in the current Internet is that a source IP address can be easily spoofed and manipulated, and *unwanted* packets can intrude an unwary host with ease (despite firewalls), which is often tricked into unintentional "accomplice" (e.g., in the case of viruses and worms), spreading attacks to many other vulnerable hosts. To combat this problem, many organizations choose to "close off" their networks via mechanisms such as VPNs or employ firewalls to block certain types of packets (e.g., based on IP addresses, ports, or packet payload), regardless of senders and their intent. Clearly, such solutions are fairly limited in their scope or effectiveness as email viruses and worms

[*] This work was supported in part by the National Science Foundation (NSF)under the grants ITR-0085824 and CNS-0435444.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 178–190, 2005.

can routinely penetrate firewalls. Furthermore, they are rather *rigid*, sometimes breaking existing applications and potentially impeding creation and deployment of new services and applications. There is still much debate in the networking research community regarding how to secure and fortify the current Internet while without jeopardizing its open architecture and end-to-end design principle.

In this paper, we propose a novel *lightweight Internet permit system (LIPS)* to provide traffic accountability through fast packet authentication for stopping unwanted packets. By unwanted packets, we mean packets *not* intended for "normal" communications between hosts, such as packets with spoofed IP addresses, generated in port scanning or worm spreading. Such packets account for, or are forerunners of, most of unauthorized accesses, intrusion, disruption, denial-of-service (DoS) attacks and other cyber threats in today's Internet. LIPS is designed as an efficient traffic authentication mechanism to filter out most of these illegitimate packets, with minor changes to current systems and negligible overheads. We implement LIPS as a small patch to the IP layer at a LIPS-aware host. When a source wants to communicate with a destination, it first requests and obtains (if granted) an *access permit* from the destination. It then inserts a destination access permit into each packet sent to the destination. Only packets with proper access permits will be accepted at the destination. This simple architecture provides a scalable and flexible framework for establishing *traffic accountability* among networks and hosts, and for securing Internet resources *without* sacrificing their open and dynamic nature. Furthermore, LIPS also simplifies and facilitates the early detection of, and timely protection from, network intrusion and attacks by requiring valid access permits before any data packets can be accepted and processed. Hence by incorporating *active monitoring* and *rapid response* mechanisms into LIPS, we can build an effective and scalable "first-line" defense to protect Internet resources from unwanted traffic.

Related Works. Many security mechanisms have been proposed to control the damage of (DDoS) attacks and trace back attacking sources. Pushback [7] treats DDoS as a congestion-control problem and requires each router to detect and preferentially drop packets that probably belong to an attack. Upstream routers are also notified to drop such packets in order that the routers resources are used to route legitimate traffic. The network capability scheme [1] inserts special tokens into packets and use routers to check these tokens along forwarding paths for restricting unwanted packets. While these routers authenticate packets and maintain *per-flow states*, destination hosts also keep *per-flow states* for authentication using hash chains. Furthermore, overlay approaches (SOS and Mayday) [5] use a *wide-area* overlay infrastructure with a *large number* of intermediate nodes to filter out attacking traffic. IP Easy-Pass [8] aims to protect real-time priority traffic Denial-of-QoS attacks at an ISP edge router by maintaining *per-flow states.* In addition, the visa protocols [3] use encryption and data signatures to authenticate a flow of packets. They require a shared key to be established between access control servers on a *per-source-destination* basis. Similarly, IPsec and VPNs establish shared keys to secure end-to-end communications with known overheads [4]. Several traceback schemes were proposed to track down attacking sources, such as IP traceback. These schemes usually require to modify intermediate routers along packet forwarding paths and are mostly for post-attack analysis. To deal with IP-spoofing, ingress filters can not stop attackers to spoof in valid address space. However, the computational overhead of public key schemes limits host performance and makes them difficult to scale to large systems.

In summary, these approaches generally either incur high computational overheads and/or heavy key management costs, or require modification to intermediate routers or broad infrastructure support. On the contrary, the proposed LIPS does not use encryption or digital signatures, hence the overheads of encryption and key management are minimized. Furthermore, LIPS is an end-to-end/edge-to-edge approach that does not require any support from *intermediate* networks. Therefore, it is easier to be deployed incrementally to a large scale.

The remainder of this paper is organized as follows. In Section 2, we present the basic concepts and constructs of LIPS, illustrate how it works and why it is useful, and discuss the related work. In Section 3, we present the design and implementation of LIPS. In Section 4, we evaluate the performance of LIPS via simulations and experiments. We conclude this paper in Section 5.

2 Basics of LIPS Architecture

The idea of LIPS is simple: every LIPS packet carries an *access permit* issued by its destination, and this permit is verified at the destination to determine whether the packet is accepted or dropped. Hence for a source to send packets to a destination, it must obtain a valid access permit first. This simple mechanism enables the destination to easily eliminate illegitimate/spoofed packets and *control who has access to it.* As only data packets with valid access permits will be accepted and processed by applications running on a destination host. Thus a malicious host cannot simply inject unwanted traffic to harm a destination host without first requesting an access permit and identifying itself to the destination. In the following, we first introduce the basic components and operations of LIPS, and illustrate how access permits are generated, exchanged, and verified. We then discuss the advantages and limitations of LIPS at the end.

LIPS Packet. LIPS is a simple extension of the IP protocol. We convert an IP packet into a LIPS packet by inserting a LIPS header into the payload field of the IP packet and changing the protocol type to 138 in the IP header[1], as shown in Fig.1. (We choose 138 as the protocol type of LIPS in our prototype implementation.) The format of a LIPS packet header is given in Fig.2, which includes four control fields, a *destination access permit (DAP)*, and a *source access permit*

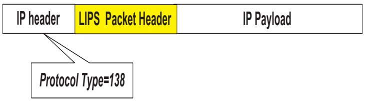
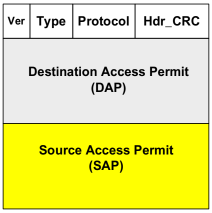


Fig. 1. A LIPS Packet

[1] To avoid the potential segmentation issue, we first perform a path MTU discovery and then set a proper MTU for the connection.

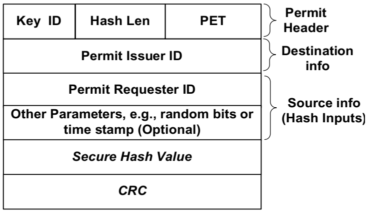


Ver | Type | Protocol | Hdr_CRC

Destination Access Permit (DAP)

Source Access Permit (SAP)

Fig. 2. LIPS Packet Header



Key ID | Hash Len | PET

Permit Issuer ID

Permit Requester ID

Other Parameters, e.g., random bits or time stamp (Optional)

Secure Hash Value

CRC

Fig. 3. LIPS Permit

(SAP). A DAP is issued by a destination to a source. It is carried in packets from the source to the destination and is verified at the destination. A SAP is issued by the source to the destination for packets on the return path. The *Ver* field holds a LIPS version number. The *Type* field specifies the type of a LIPS packet, such as a permit request, a permit reply, or a LIPS data packet. Since we replace the IP protocol type in the IP header to 138 when we translate an IP packet into a LIPS packet, we use the *Protocol* field to hold the protocol type in an original IP packet such that we can restore it after the LIPS packet is accepted at a destination. The *Hdr_CRC* field is a simple CRC for a LIPS packet header.

Access Permit. An access permit is constructed using *keyed* message authentication code (MAC) [6] at a LIPS-aware host. This MAC is generated through a secure hash function with two inputs: a *plain* hash message (chosen by a permit issuer and carried in an access permit in plain text) and a *secret* hash key held by the issuer.

As shown in Fig.3, an access permit includes five parts: a *permit header*, a *permit issuer's ID*, a *permit requester's ID* (plus optional parameters), a *secure hash value*, and a *CRC checksum* of the secure hash value. The permit header contains an index (*Key ID*) of a secret key used for this permit at its issuer, a hash length (*Hash Len*) that specifies the length of the secure hash value in this permit, and a permit expire time (*PET*) that defines the effective duration of this permit. The length of secure hash value can be adjusted from 64 bits to 128 bits depending on the permit issuer's security requirements. The source information (denoted as M) includes, e.g., the source IP address and several optional security parameters (e.g., a random number used to deal with permit-replay attacks). It is used by the issuer as the input plain hash message to a secure hash function to compute a message digest, $H(M, K_t)$, where $H()$ is a secure hash function, e.g., HMAC-MD5, and K_t is a secret key of the permit issuer at time t. For ease of exposition, we use the source IP address as M in the following presentation. Given that the hash length in the permit header is l, the secure hash value of a permit is the first l bits of the message digest. For example, we can choose the first 64 bits of a 128-bit HMAC-MD5 digest as a secure hash value. This hash value will be used for validating the permit. Note that the hash value is *specific* to the requester and valid only for a certain period of time, as K_t is changed over time. Without knowing the secret key, it is very difficult to forge a permit.

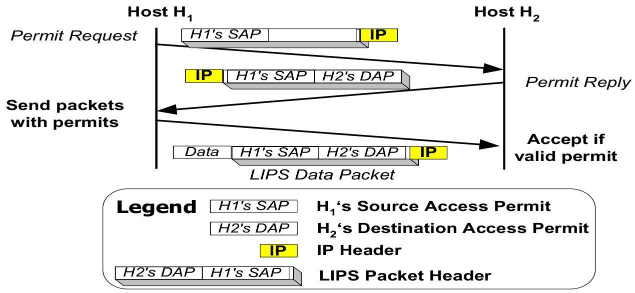


Fig. 4. Message Exchanges in Setting up an Access Permit

Exchange of Access Permits between a Source and a Destination. We use a simple example to illustrate how access permits are set up between two LIPS-aware hosts. As shown in Fig.4, when a source host H_1 wants to communicate with a destination host H_2, H_1 first sends a *permit request* to H_2. This request carries H_1's SAP in the request. The SAP contains a secure hash value generated based on a *secret* hash key of H_1 and a plain hash message about H_2 (e.g., H_2's IP address). *Note that not all hosts will be allowed to access* H_2*.* A *security policy* at H_2 is checked to determine if H_2 accepts this permit request[2]. If it does, it generates an access permit (H_2's DAP for H_1), containing a secure hash value generated based on a secret hash key of H_2 and a plain hash message about H_1 (e.g., H_1's IP address). Then H_2 sends the permit (as the SAP) in a *permit reply* message back to H_1, using H_1's SAP (attached in the permit request from H_1) as the DAP.

H_1 will only accept a permit reply that carries a valid DAP, namely, a SAP that it issues in an earlier permit request. This is done by computing a secure hash value using the plain hash message carried in the DAP and a secret key pointed by the key index. If this hash value matches the secure hash value carried in the DAP, H_1 accepts this reply and caches the SAP of the packet (i.e., H_2's DAP) into a *permit cache*. For the subsequent data packets sent to H_2, H_1 puts H_2's DAP and its own SAP into the LIPS headers of these packets. When H_2 receives a LIPS packet from H_1, it verifies the DAP of the packet and accepts it only if the DAP is valid.

Table 1. Permit Cache

Destination IP address	Flag	Destination Access Permit
172.10.10.1	1	xxxxxxxx
129.128.128.1	3	NULL

Permit Cache. Each LIPS host maintains a permit cache with a format shown as Table. 1. A cache entry contains a destination IP address, a *Flag*, and a destination access permit. For incremental deployment, the cache not only holds permits

[2] An example policy may be only accepting permit requests from a local domain, e.g., accepting packets from (secure) proxy servers when communicating with hosts in other domains. This will automatically eliminate port scanning and worm spreading packets from other domains, i.e., an attacker outside a domain cannot discover vulnerabilities via scanning and a worm cannot propagate across domains through random probing. Damages are localized.

for LIPS-aware hosts, but also tracks non-LIPS hosts (allowed by security policies), using the destination IP address as its primary index. The flag is used to distinguish the state of a cache entry: flag = 0, indicating that the entry is in initialization, namely, a permit request has been sent to the destination but the reply has not been received yet; flag = 1, a valid permit for the destination is available; flag = 2, the destination permit has expired; and flag = 3, the destination does not supports LIPS.

Key Management. Each LIPS host maintains a secret key pool of, say, 256 keys. Each key is uniquely identified by a key index. When a host generates an access permit, it randomly chooses a key from its key pool and records the key index in the *Key ID* field of a permit header. When a host verifies an access permit, it retrieves a key using the *Key ID* of the permit header. Note that in LIPS a key pool is *not shared* with any other hosts, and *no key exchange* among hosts is required, contrary to the complex key establishment procedures in other approaches. Hence the overhead of LIPS key management is minimized.

Why LIPS. We conclude this section by illustrating how LIPS can be used as a first-line defensive and preventive mechanism to protect hosts from unwanted traffic. First, LIPS introduces *traffic-origin accountability* into the Internet and enables destinations with the ability to deny access and stop unwanted traffic from untrusted hosts. Second and perhaps more importantly, *LIPS facilitates and simplifies the tasks of detecting unauthorized intrusion and attacks* by forcing malicious hosts to first requesting access permits and identifying themselves to the intended targets before launching an offense, as we illustrate below. Clearly, since a source must obtain a valid access permit before it can send a packet to a destination, illegitimate/spoofed packets will be automatically filtered out. Cyber attacks such as worms generally rely on port-scanning to identify vulnerable hosts, tricking them to execute the malcode carried in the payload, and thereby compromising them as stepping stones or unintentional accomplices to further spread attacks. Similarly, DoS attacks rely on target hosts to expend valuable resources by unnecessarily processing bogus service requests. Since a malicious host cannot simply inject unwanted traffic to harm a LIPS-protected destination without first requesting a permit and identifying itself to the destination[3], we can better defend such attacks by simply *detecting anomalies in permit request traffic*. For example, a sudden/unusual surge of permit requests to one or more hosts in a protected network signifies suspicious activities. In particular, when implemented in the gateway mode (introduced in the next section), a source zone can detect attacks originating from malicious or worm/virus-affected hosts within its zone and quarantine them by denying (host-specific) request permits to their target destinations. Hence combined with network intrusion mechanisms, LIPS can form an effective first line of defense against cyber attacks by stopping unwanted

[3] An attack may flood a destination, but its packets will be simply dropped and can not harm applications as today's worms. We will discuss the prevention of flooding attack in Section 4.

traffic. In summary, LIPS is designed to localize spoofing and associated attacks, restrict worm spreading, stop random probing and reflection attacks, assist IDSs in significantly reducing their load and providing cross-domain feedbacks, and protect important servers and their incoming links.

3 LIPS Design and Implementation

For incremental deployment and scalability, we design LIPS operating in two modes. The basic LIPS works in a *host mode*, in which a LIPS-aware host directly communicates with another LIPS-aware host as introduced in the previous section. The LIPS host mode is used as an incremental approach to deploy LIPS when a few LIPS-aware hosts directly communicate with each other in a small scale, and it is also used for communications within a zone under the gateway mode when LIPS is deployed in a large scale. We refer readers to [2] for the details of LISP host mode. Here we mostly introduce the LISP *gateway mode*.

We organize LIPS-aware hosts into *secure zones* based on their network administrative domains or zones. We use *zone access permits* to authenticate *inter-zone* packets, and use *host access permits* (as in the host mode) to authenticate *intra-zone* packets. Each zone has a *permit server (PS)* to manage inter-zone permits and a *security gateway (SG)* to validate inter-zone packets based on inter-zone permits. Once an inter-zone permit is established between a pair of zones, the subsequent communications between them will take advantage of this permit and avoid repeatedly setting up inter-zone permits. As a result, we not only reduce permit setup delays but also significantly reduce inter-zone permit exchange traffic. Furthermore, we propose a unique and simple *permit-mutation* method to transform zone permits and host permits back and forth such that *not only security gateways do not need to keep per-flow states but also zone permits are not revealed to hosts.* Another advantage of permit-mutation is to localize damage caused by potential attacks as discussed later. In each zone, LIPS-aware hosts still directly communicate with each other as in the LIPS host mode.

Permit Server, Intra-zone and Inter-zone Permit Setup Protocol. As show in Fig.5, host H_1 in zone Z_1 wants to access host H_2 (e.g., a protected application server) in zone Z_2. PS_1 is the permit server of zone Z_1, and PS_2

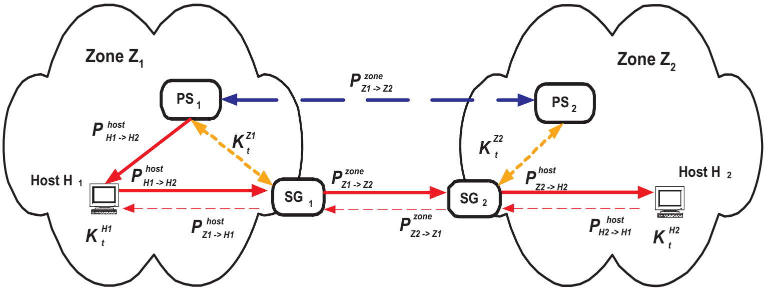


Fig. 5. Illustration of LIPS Gateway Mode

is the permit server of zone Z_2. Zone Z_1 and zone Z_2 are protected by security gateways SG_1 and SG_2, respectively, which authenticate both ingress and egress traffic originating from and destining to trusted hosts in these zones. To obtain a permit to access remote host H_2, H_1 authenticates itself to its local permit server PS_1 (e.g., a local authentication scheme such as Kerberos). PS_1 assists H_1 to obtain an access permit to H_2. Under a two-tiered model, we divide the packet forwarding path from a local host H_1 to a remote host H_2 into three segments: from H_1 to SG_1, from SG_1 to SG_2, and from SG_2 to H_2. Correspondingly, we use three access permits at each of these segments for packet authentication: an intra-zone host access permit $P^{host}_{H_1 \to H_2}$, an inter-zone permit $P^{zone}_{Z_1 \to Z_2}$, and another intra-zone host access permit $P^{host}_{Z_2 \to H_2}$. We introduce the setup protocols for these permits in the following.

Each PS is assigned a *zone ID*. In this prototype design, we simply choose the IP address of a PS as its zone ID since inter-zone permit requests and replies will be exchanged between PSs. We use this zone ID to generate a zone access permit as follows. In response to a permit request from a trusted host, the local PS passes the request to the corresponding (authoritative) PS in the remote secure zone[4], together with its zone ID and other necessary credentials. If the access is allowed, the remote PS will generate a *zone access permit* (or *zone permit* in short) based on the local PS's zone ID. Hence the access permit is *source-zone specific*. The remote PS returns the zone permit to the local PS together with its *own* zone ID. Instead of directly passing the zone permit to the requesting host, the local PS creates a new *host access permit* (or *host permit* in short) by adding some "random" value generated based on the source and destination IP addresses as explained in the following. This *mutation* of a zone permit into a host permit makes the host permit *specific to both source host and destination host*, thereby rendering it difficult to be spoofed by other hosts.

Zone Access Permits. Zone access permits are generated in the same fashion as host access permits but use a zone ID as a plain hash message. For a packet, let use IP_1 to denote its source IP address of host H_1 in zone Z_1, and use IP_2 to denote its destination IP address of host H_2 in zone Z_2. Let IP_{PS_1} be the IP address of a requesting PS (as a zone ID), and $K_t^{Z_2}$ be a *secret key* maintained by the queried PS_2 at time t. Then the secure hash value of the zone permit is $P^{zone}_{Z_1 \to Z_2} = H(IP_{PS_1}, K_t^{Z_2})$, where $H()$ is a secure hash function, and the CRC checksum of the permit is computed on $P^{zone}_{Z_1 \to Z_2}$. As explained in the following, the CRC checksum is used to verify the validity of the permit after the *permit de-mutation* for outbound packets. Note that the generated permit is *specific* to the requesting zone, and is valid only for a certain period of time, as $K_t^{Z_2}$ changes over time. Without knowing $K_t^{Z_2}$, it is very difficult to forge a zone permit.

[4] For a PS to find an authoritative PS of a domain, we add a simple resource record at the DNS of a domain such that a PS can find another PS through a simple DNS query, based on a simple name convention. We assume that DNSsec will solve security issues related to current DNS and so we will not discuss DNS security in this paper.

Mutation of a Zone Permit to a Host Permit. Given the zone permit $P^{zone}_{Z_1 \to Z_2}$, the requesting PS *mutates* it into a host permit $P^{host}_{H_1 \to H_2}$ using the IP address of the requesting (source) host, IP_1, and the IP address of the queried (destination) host IP_2. Let $K^{Z_1}_t$ be a *secret key* maintained at the requesting PS at time t. We construct a host permit, $P^{host}_{H_1 \to H_2} = P^{zone}_{Z_1 \to Z_2} \oplus H(IP_1, IP_2, K^{Z_1}_t)$. Note that the host permit $P^{host}_{H_1 \to H_2}$ is only valid for the source H_1 to access the destination H_2 for a certain period of time. Again, without knowing the secret key $K^{Z_1}_t$, it is also very difficult to forge a host permit. The host permit is essentially the same as the zone permit, with the secure hash value $P^{zone}_{Z_1 \to Z_2}$ replaced by $P^{host}_{H_1 \to H_2}$. Note that the CRC checksum is *not* re-computed.

Host and Gateway Operations. At both source and destination domains, we establish lightweight packet authentication mechanisms for verifying and filtering packets based on host and zone access permits.

Host Operations. In the gateway mode, we install a *host authentication layer (HAL)* at each host. During its initialization, a HAL authenticates itself to its permit server and security gateways, e.g., via a local authentication scheme such as Kerberos. This authentication only occurs once during its initialization. In the meantime, it also issues *host access permits* to its PS and its SG for authenticating permit replies and LIPS data packets from them, e.g., host H_1 issues permit $P^{host}_{Z_1 \to H_1}$ to PS_1 and SG_1.

The HAL layer at an end host x intercepts each outbound packet and then looks up its permit cache based on the destination IP address of the packet. If a destination access permit is found, it is attached the permit to the packet. In addition, the host will attach its source access permit generated using its local zone ID IP_{PS_i}, $P^{host}_{Z_i \to x} := H(IP_{PS_i}, K^{H_x}_t)$, where $K^{H_x}_t$ is a secret key kept by the host at time t. This source access permit is used for authenticating packets from the security gateway to the host. For each *incoming* packet, the HAL checks the validity of the destination permit using the *destination zone ID* (carried in the permit) and its own secret key. (It is the *reverse* operation of generating the source access permit in the above). The packet is accepted only if it passes the verification. In this case, the source access permit is *cached* in the permit cache (with a timer appropriately set, in a manner similar to the ARP table used for IP and MAC translation).

Gateway Operations. The *gateway authentication* layer (GAL) is a LIPS realization at a SG, which is a small patch to the IP layer. For *outgoing packets*, the SG is responsible for ensuring that they are authorized to access the protected remote zones and hosts. To verify this, it uses the source IP address IP_1, the destination IP address IP_2, and the destination access permit $P^{host}_{H_1 \to H_2}$ (carried in the packet) to first compute $X := P^{host}_{H_1 \to H_2} \oplus H(IP_1, IP_2, K^{Z_1}_t)$, where $K^{Z_1}_t$ is the secret key that the SG shares with the local PS. It then generates the checksum on X. If the computed checksum *does not* match the checksum carried in the destination access permit, the authentication fails and the packet is dropped. Otherwise, the secure hash value in the destination access permit is replaced by

X (note that $X = P^{zone}_{Z_1 \to Z_2}$), and thus the destination access permit is *de-mutated* back to the original zone access permit issued by the destination zone. Furthermore, the (host) source access permit of H_1, together with the source host IP address, is cached in the LIPS permit cache at the gateway. In addition, the gateway will replace the (host) source access permit in the packet with a new (*zone*) source access permit, $P^{zone}_{Z_2 \to Z_1} := H(IP_{PS_2}, K^{Z_1}_t)$, where IP_{PS_2} is the IP address of PS_2 as the *destination* zone ID. $P^{zone}_{Z_2 \to Z_1}$ is used to authenticate packets from the destination zone Z_2 on the reverse path.

For packets entering a destination zone, the security gateway is responsible for verifying that they carry proper zone access permits. This is done by checking to see whether the destination permit carried in an incoming packet, $P^{zone}_{Z_1 \to Z_2}$, is valid. If this verification fails, the packet is discarded. Otherwise, the packet is allowed to enter the destination zone. Using the destination IP address IP_2, the gateway looks up its permit cache and replaces the destination zone permit with the corresponding destination host permit. Depending on whether the destination host is a trusted host (e.g., a server) in a *protected* (e.g., secluded) network, or a client host in a less secure environment, the gateway may replace the source *zone* access permit, $P^{zone}_{Z_2 \to Z_1}$, with a mutated source *host* access permit, $P^{host}_{H_2 \to H_1} := P^{zone}_{Z_2 \to Z_1} \oplus H(IP_1, IP_2, K^{Z_2}_t)$. In the former case, for scalability this operation is *optional* so that trusted servers and other high-performance hosts in protected networks only need to maintain zone-level access permits. In the latter case, this operation would prevent other untrusted hosts to eavesdrop and forge (zone) access permits. A detailed illustration of operations in LIPS gateway mode can be found in [2].

Advantages of LIPS Design and Implementation. The design and implementation of LIPS have the following several salient advantages. As noted earlier, a key feature of LIPS is that *no secret* is shared across network domains, which makes the architecture more scalable and flexible. Packet authentication is performed using only information carried in (the LIPS header of) a packet and *secret keys* held *locally* by security gateways and hosts. Thus packet operations can be done efficiently. Furthermore, our architecture is purely *edge-to-edge* (or "end-domain-to-end-domain"), as it does not require any *intermediate* networks for assistance. It is also *incrementally* deployable: only those hosts that need to be secured have to be patched with simple protocol enhancement, and to be placed "behind" security gateways for authentication and protection. In addition, *no* modification to applications is required. Through separate *zone-level* and *host-level* access permits, we isolate "bad" packets originating in one's own zone from those outside, and limit the abilities of attacks to mostly "man-in-the-middle" *replay* attacks by "sniffing" permits. Permit- or Packet-Replay attacks can be mitigated by including, e.g., sequence no. or random bits, in access permits. By augmenting LIPS with active monitoring and rapid response defense mechanisms, we can quickly detect and throttle such attacks (e.g., by detecting duplicate access permits, and adjusting timed keys). With such mechanisms, replay attacks will have very *localized* effect, with only "sniffed" hosts/domains being affected, due to the host-specific/domain-specific feature of access permits.

4 Performance Evaluation

We have evaluated the basic overhead of the LIPS itself, and examined the effectiveness of LIPS in protecting server resources. In the following, we present these results briefly. Readers can find more details and results in [2].

Overall Overhead of LIPS. We conducted experiments to examine the overall overhead introduced by our LIPS implementation on Linux platforms in data transmission, compared with IP. In these experiments, we used Iperf to send an CBR UDP flow from a host to another via a dedicated 100Mbps link. When the CBR rate is lower than 100Mbps, there are almost no differences between the transmissions with or without LIPS. Table 2 shows the Iperf measurements when the CBR rate is 100Mbps. Even in this stress test, the difference between the transmission bandwidth with LIPS and that with IP is negligible small (3%). In our experiments, we also measured the delays of key LIPS operations including HMAC computation, permit lookup, and mutation. On a common Linux platform with 2.8MHz Pentium, we can authenticate around 640 Mbps [2], far beyond a common user's requirement.

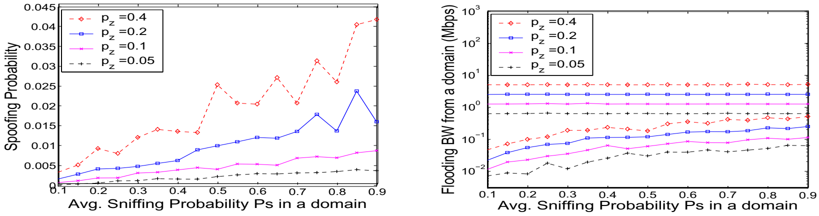
Effective LIPS Protection. We further use analytical models to show how LIPS helps stop DoS attacks from two aspects: the chances for zombies to start DoS floods and the probabilities of successful attacks. We focus on the replay of host permits in LIPS domains because it is rather difficult to gain access to inter-domain links to sniff a domain permit. The real time and host-specific nature of LIPS permits dramatically increases the difficulty to generate attacking traffic. Furthermore, we design a fast response mechanism to quickly stop floods. Therefore, it is extremely difficult to bring down a LIPS-protected target.

We first examine the spoofing chances for a zombie in a LIPS domain. Under LIPS, to sniff host permits for spoofing, a zombie must have access to the path to a destination in real time. We define p_z as the probability that a host is compromised as a zombie in a domain, and p_s as the probability that a zombie can sniff a valid permit to a target in the domain. Assume that a legitimate host communicates with a target server as a Poisson process. As shown in Fig.6(a), given different p_z and p_s, the spoofing probabilities for zombies under LIPS are far lower than 1, the chance under IP.

Consequently, LIPS dramatically suppresses zombies' capabilities to launch flooding attacks to a target. Assume we have a domain of 100 hosts; those hosts communicate with a remote server as Poisson processes; the mean flow rate of a legitimate session is 128Kbps.

Table 2. Comparison: with and without LIPS over a dedicated link

	Effective Bandwidth	Loss Rate	Jitter
With LIPS	90.7 Mbps	0.005%	0.025ms
W/O LIPS	93.7 Mbps	0.005%	0.022ms



(a) Spoofing Probabilities in LIPS for a zombie

(b) Comparison of Aggregate Spoofing Bandwidth from a domain: IP vs. LIPS

Fig. 6. LIPS significantly reduces spoofing chances and restricts flooding capability

Fig.6(b) shows that the aggregate flooding bandwidth to the server, which can be generated by zombies in the domain. The top four lines are flooding rates under IP with various p_z, while the bottom four lines are flooding rates under LIPS with the same conditions. Note that the Y-axis is in a log scale. Clearly, it is very difficult for zombies to generate sufficient traffic to flood the server under LIPS; while it is fairly easy under IP. In addition, our study also shows that an attacker needs about 10^4 to 10^5 domains as the above to flood a LIPS-protected 1 Gbps link with over 100% unwanted packets. It is extremely difficult for an attacker to collect such huge amount of resources.

Furthermore, we use a simple Stochastic Knapsack framework [5] to model a DoS attack to a protect incoming link of a target. We use C to denote the total amount of incoming bandwidth available. Assume legitimate flows (or attacking flows) have an exponential arrival rate with a mean of λ_l (or λ_a), a bandwidth requirement b_l (or b_a), and an exponential service time with a mean of μ_l (or μ_a). The system admits an arrival whenever bandwidth available. In this model, the probability of a successful DoS attack is the blocking probability corresponding to the legitimate traffic, defined as $P_b = 1 - \frac{\sum_S (\rho_l^{n_l}/n_l!)\cdot(\rho_a^{n_a}/n_a!)}{\sum_{S'} (\rho_l^{n_l}/n_l!)\cdot(\rho_a^{n_a}/n_a!)}$, where S is the set of cases that an arriving legitimate flow can be admitted, and S' is the set of cases that either a legitimate flow or an attacking flow is admitted; in each case, n_l is the number of legitimate flows admitted, and n_a is the number of attacking flows admitted; and offered load $\rho_l = \lambda_l/\mu_l$, $\rho_a = \lambda_a/\mu_a$. Fig.7 shows the blocking probability of legitimate flows as we increase the load of attacking traffic. To block 90% of legitimate traffic, the attacking load has to be 1000 times heavier than the legitimate traffic.

We also design an inter-domain collaboration scheme to stop permit-replay flooding in LIPS domains and take care of zombies across domain [2]. Since zombies have to generate high attacking load to launch successful attacks, it is easy to identify them and then isolate these zombies through a local defense mechanism. Furthermore, we can also identify and deal with zombies through automatic inter-domain collaborations. Fig. 8(a) and Fig. 8(b) show the effectiveness of our fast response scheme: to protect incoming links with various

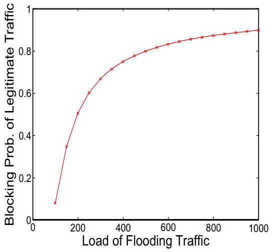
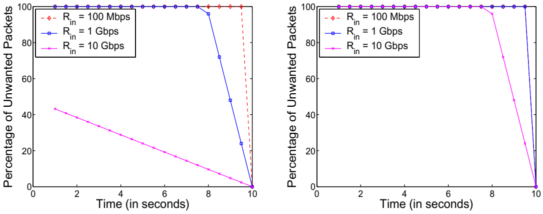


Fig. 7. Blocking Probability of legitimate flows as the attacking traffic load increases



(a) 1,000 replaying sources: 100 sources in each of 10 domains

(b) 10,000 relaying sources: 100 sources in each of 100 domains

Fig. 8. Static Attacks: Delays to shut off all replaying sources

capacities, we can quickly shut off these replaying source in 10 seconds for a large number of replaying sources.

5 Conclusions

LIPS is a simple packet authentication mechanism which provides traffic origin accountability for stopping unwanted traffic. We presented the basic design of LIPS and a prototype implementation on Linux platform. Our analytical, simulation and experimental results show that LIPS is capable of stopping unwanted packets with negligible overheads. Currently, we are incorporating active monitoring and rapid-response defense mechanisms into LIPS for further improving its security and performance.

References

1. T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *Hotnets 2003*, Nov. 2003.
2. Y. Dong, C. Choi, and Z. l. Zhang. A lightweight permit system for stopping unwanted packets. *Technical Report, http://www.ee.hawaii.edu/~dong/papers/LIPS_report.pdf*, July, 2004.
3. D. Estrin and et. al. Visa protocols for controlling inter-organization datagram flow. *In IEEE Journal on Selected Areas in Communication*, May 1989.
4. G. Hadjichristo, N. Davis IV, and C. Midki. Ipsec overhead in wireline and wireless networks for web and email applications. *In Proc. of IEEE IPCCC*, Apr. 2003.
5. A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. *In Proc. of ACM SIGCOMM*, Aug. 2002.
6. A. Menezes, P. Oorschot, and S. Vanstone. Handbook of applied cryptography. *CRC Press, ISBN: 0-8493-8523-7*, 1996.
7. K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. *Proc. of ACM SIGCOMM 2001, San Diago, CA.*
8. H. Wang, A.Bose, M.Gendy, and K.Shin. IP Easy-pass: Edge Resource Access Control. *In Proc. of IEEE INFOCOM*, 2004.

Victim-Assisted Mitigation Technique for TCP-Based Reflector DDoS Attacks

Basheer Al-Duwairi and G. Manimaran

Department of Electrical and Computer Engineering,
Iowa State University, Ames, IA 50011, USA
{dbasheer, gmani}@iastate.edu

Abstract. This paper develops the concept of victim-assistance for denial of service (DoS) mitigation. The proposed concept is utilized within a simple, yet effective scheme designed for mitigating TCP-based reflector DoS attacks. The proposed scheme, called SYN number based filtering (SNF), takes into account the TCP's connection establishment behavior and the inherent features of the attack itself. The main idea of the SNF scheme is to restrict the choice of the initial sequence numbers of SYN packets to certain pattern, such that corresponding SYN-ACK packets can be validated at the ISP's perimeter. We evaluate the proposed scheme through analytical studies for classical and advanced attacks using two performance metrics, namely, the false positive and false negative rates. Our analysis shows that the proposed scheme offers low false positive and false negative rates. In addition, we identify several research problems based on the proposed concept.

1 Introduction

Recently, reflector-based denial of service (RDoS) attacks are being used by attackers frequently to affect the availability of high profile servers [8, 13]. In these attacks, a large number of compromised computers under attackers control (the slaves) are instructed to continuously send request packets to a set of Internet reflectors (an Internet reflector is an IP host that will reply to any request packet). The source address of each of these request packets is spoofed to be the same as the address of the targeted site. As a result, the reflectors send their replies to the given address causing packet flooding at that site. Using Internet reflectors complicates the problem of DoS attacks. Researchers are more concerned about these attacks because attack packets (reply packets originated from the reflectors themselves) carry legitimate IP source addresses making it difficult to trace original attack sources. Also, because these attacks are usually characterized by an amplification factor that increases their intensity.

The analysis presented in [13] shows that RDoS attacks are feasible in variety of request/reply based protocols, including TCP, UDP, ICMP, and DNS. For example, in Smurf attacks [3], the attacker sends ICMP echo requests (pings) to

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 191–204, 2005.

the broadcast address of a network, so the victim is hit by many more packets. The Fraggle (UDP packet magnification) attack uses UDP echo packets in the same fashion as the ICMP echo packets. In TCP-based RDoS attacks [8], attackers may take advantage of the availability and connectivity of large number of Internet reflectors to coordinate a highly distributed DoS attack. This can be done by abusing the TCP protocol in the following way: An attacker, A, selects a set of Internet reflectors $R=\{R_1,R_2, ...,R_n\}$. It then sends low rate faked SYN packets to each of these reflectors with a spoofed source address equals to that of the final target, V. For each received SYN packet, the reflectors reply with a SYN-ACK packet to the given address, V. Therefore overwhelming the victim site, V, by high aggregate rate of SYN-ACK packets.

Mitigating RDoS attacks is extremely important issue. In this paper, we develop the concept of victim-assistance and use it in the context of a novel scheme to mitigate TCP-based RDoS attacks. This type of attacks has been encountered repeatedly in recent years. For example, attacks against GRC.com were reported and analyzed in [8]. In addition, many TCP-based reflector attacks were captured in the traffic traces taken at Los Nettos ISP network [10]. Its special importance is due to the following reasons: (1) different than other types of RDoS attacks, *TCP-based RDoS cannot be mitigated by blind filtering of attack packets, because such solution would prevent the victim itself from establishing any TCP connection*[1], (2) attack amplification is achieved through multiple retransmission of SYN-ACK packets by the reflectors themselves after each time out, (3) attackers are attracted by the fact that TCP carries 95% of today's Internet traffic and 80% of the total number of flows in the Internet [12], and (4) the fact that any general purpose TCP connection-accepting Internet server could be used as a packet reflection server, which provides attackers with large pool of servers to be used as reflectors.

The proposed scheme, called SYN Number-based Filtering (SNF), is based on restricting the choice of the initial sequence number (ISN) of TCP connection establishment requests initiated by the victim, such that legitimate reply packets can be validated at the ISP (Internet Service Provider) perimeter. The ISN restriction is achieved by requiring it to contain a secret pattern, C_s, known only to the victim and to the ISP's edge routers. We evaluate the proposed scheme by first assuming a classical attack in which attackers do not react to the defense scheme. Then, we take into account the ability of the attacker to perform advanced attacks in which attack packets are generated with ISNs that contain the C_s used by the victim. A simple stochastic analysis are conducted to gain an insight on the performance of the proposed scheme. The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 develops the concept of victim-assistance. Section 4 describes the proposed scheme. Finally, conclusions and future research problems are outlined in section 5.

[1] Examples of servers that could be targets of this type of attacks include servers that initiate TCP connections frequently, such as FTP server, proxy server, and Socks server.

2 Related Work

In general, RDoS attacks can be defeated either by filtering the reflected attack packets (which hold valid IP source addresses), or by solving the origins of the problem (i.e., filtering IP packets with spoofed source addresses). In this section, we discuss the main research efforts that addressed the RDoS explicitly, and we review some of the research efforts that targeted the filtering of spoofed IP packets.

2.1 Detection and Prevention of RDoS Attacks

In [13], filtering of different types of reply attack packets that share certain attributes, such as the destination port number and IP destination address, was considered. However, the issue of collateral damage (i.e., filtering legitimate replies as well) was not addressed. In [15], a distributed approach for detecting reflector attacks was proposed. The approach is based on sharing beliefs among potential reflectors if any abnormal traffic is observed, such that the reflectors become aware that they are being used in a RDoS attack, and consequently start ignoring incoming request packets that have source address equals to the victim's address. Clearly, this approach cannot be deployed in practice because there is no way by which certain reflector knows the group of reflectors participating in ongoing attack, such that it can share its belief with them. Even if attack detection is possible among set of reflectors, there is no mechanism by which reflectors can distinguish attack packets from legitimate packets. Moreover, it is possible for the attackers to abuse the scheme by sharing their own beliefs with many other innocent reflectors in order to drop legitimate requests received by them.

2.2 Filtering of Spoofed IP Packets

Generally, filtering of spoofed IP packets can be done at the source network, intermediate routers, or at the destination network. For example, in ingress filtering [7], routers are configured to block packets that arrive at the edge router of the source network with illegitimate source addresses. Obviously, the major problem with this approach is that it may violate some existing setups and protocols such as Mobile IP and multi-homing. It is also difficult to convince ISP administrators to support ingress filtering because the benefit is not felt directly by the deploying ISP. The proposed scheme is different in this aspect because an ISP network will directly benefit from its deployment. Another example is the SAVE protocol [11], which is designed to provide routers with the information needed for source address validation. The main problem of this protocol is that legitimate packets may be filtered even in the absence of an attack. This is due to routing instability which leads to errors in the source address validation tables maintained at the SAVE enabled routers. In general, such schemes require large scale deployment to prevent IP source address spoofing efficiently.

Hop count filtering [9] is a simple approach to drop spoofed packets at the destination network. It is based on observing that the distance travelled by a spoofed packet is usually different than that travelled by a packet originated from the actual spoofed source. Therefore, attack packets can be distinguished and dropped directly. The main drawback of this approach is the need to keep up to date database of source addresses and their distances. This might be difficult due to route changes. Also smart attackers may spoof IP addresses that never communicated with the given reflector such that it cannot judge about the validity of these packets.

3 The Concept of Victim-Assistance

It is well known that a DoS attack against an end-system has a direct impact not only on that system, but also on the network in which the targeted system is located. This is due to attack traffic aggregation near the victim which leads to network bandwidth exhaustion. Therefore, it is of primary importance to filter attack traffic as far away from the victim as possible. In this context, the placement of DoS detection and mitigation modules involves several tradeoffs that need to be considered. (Modules refer to devices together with the required software).

The Tradeoff Introduced by the Placement of DoS Detection and Mitigation Modules. The placement of the modules is of primary importance as it involves a tradeoff between efficiency and practicality. This is due to the fact that DoS attacks consume both network and end-system resources. At one extreme, these modules can be placed at the victim itself. At another extreme, these modules can be placed at the ISP's perimeter. Between the two extremes, the placement of DoS detection and mitigation modules can be along an optimal boundary defined by the victim. The following discussion explains the tradeoffs introduced in each case.

- *At the victim:* With respect to DoS detection, the victim is the best location for fast and accurate attack detection in most attack scenarios. With respect to DoS mitigation, the victim is also the best location from deployment point of view, because the DoS mitigation module would be installed at a single system (i.e., at victim) which has complete knowledge about the attack, and, in most cases, it can easily classify incoming traffic as attack or legitimate. Unfortunately, this ability does not alleviate the effect of attack packets as they consume significant resources by the time they are identified.
- *At the ISP's perimeter:* With respect to DoS detection, edge routers along the ISP's perimeter cannot detect the existence of an attack because the amount of attack traffic seen by each edge router individually is very small as compared to that seen by the victim. However, with respect to DoS mitigation, installing the DoS mitigation modules far away from the victim provides protection for the network as well as for the targeted system. It is preferred to install the DoS mitigation modules at the ISP's edge routers to form a line of defense at the ISP's perimeter, because (1) any traffic destined to the victim has to pass through one of the ISP's edge routers (2) edge routers are usually computationally capable and can perform the task of packet filtering. However, the major problem with this approach is that edge routers

cannot perform accurate traffic classification by themselves due to the lack of knowledge about the ongoing attack and how to make a distinction between attack packets and legitimate packets.

- *At an optimal boundary:* Optimal boundary is a conceptual term that refers to the periphery around the victim where it is possible to protect against end system exhaustion as well as bandwidth exhaustion. Such periphery is located somewhere between the ISP's perimeter and the victim itself. The DoS detection/mitigation modules installed along that periphery can (1) detect attacks that are difficult to detect at the ISP's perimeter (2) perform more efficient DoS mitigation as compared to mitigation at the ISP's perimeter. Although such placement policy achieves the best of both worlds (i.e., protection from bandwidth exhaustion with accurate detection and efficient mitigation), it has several practical limitations that prevent its deployment. For example, the conceptual optimal boundary is victim-oriented. Moreover, it requires modification at core routers. Therefore, raising deployment concerns.

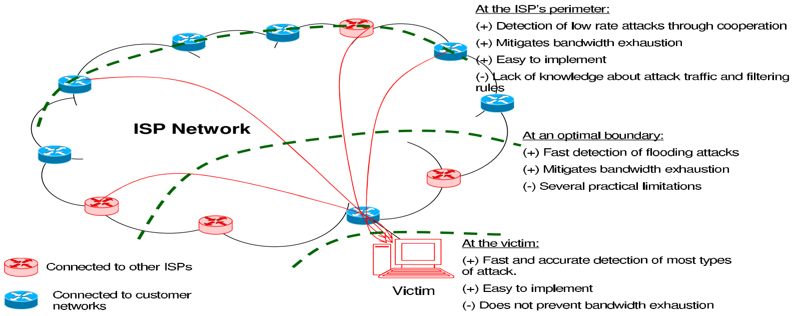

At the ISP's perimeter:
(+) Detection of low rate attacks through cooperation
(+) Mitigates bandwidth exhaustion
(+) Easy to implement
(-) Lack of knowledge about attack traffic and filtering rules
ISP Network
At an optimal boundary:
(+) Fast detection of flooding attacks
(+) Mitigates bandwidth exhaustion
(-) Several practical limitations
At the victim:
(+) Fast and accurate detection of most types of attack.
(+) Easy to implement
(-) Does not prevent bandwidth exhaustion
Victim
Connected to other ISPs
Connected to customer networks

Fig. 1. The tradeoff introduced in the placement of DoS detection and mitigation modules within an ISP network

Fig. 1 depicts the tradeoff introduced by the placement of the DoS detection/mitigation modules. Obviously, *performing DoS mitigation at the ISP's perimeter provides protection from both bandwidth and end system exhaustion.* In fact, the perimeter-based DoS mitigation architecture is not new as it has been used in previous work (e.g., [2, 5]). However, different than earlier research, we introduce the concept of *online* perimeter-based DoS mitigation wherein edge routers of the ISP network take advantage of the inherent features of the ongoing attack to perform per-packet classification and filtering. In this context, ISP's edge routers should be supplemented by special mechanisms in order to perform online attack mitigation. The concept of victim-assistance is proposed to achieve this objective. Victim-assistance refers to the direct role of the victim in identifying attack traffic before reaching its target. The mitigation scheme at ISP's edge routers requires the victim to cooperate with them in a manner that leads to accurate classification of incoming traffic. In this context, several research problems can be identified. For example, what information should be provided by the victim? How this information is to be communicated to edge routers? When to activate/terminate the mitigation scheme and which edge routers? In fact, the answers to these questions depend on the attack type which can be determined by the victim itself. In the next section, we illustrate the use of this concept in mitigating TCP-based RDoS attacks.

4 SYN-Number Based Filtering (SNF)

TCP connection establishment procedure is characterized by its deterministic nature regarding the type and content of messages exchanged between the communicating parties. This forms the basis of our scheme for defense against TCP-based RDoS attacks. Theoretically, an incoming SYN-ACK packet (destined to the victim) can be validated by inspecting its sequence number to see if it matches the ISN of the corresponding SYN packet plus one. Although such validation could be done very effectively at the victim itself, it would not be useful to stop bandwidth exhaustion. At the same time, it is more challenging to perform validation at the edge routers of the ISP network that contains the victim, because it would be necessary to maintain state information about each SYN packet sent by the victim.

The proposed scheme, called SYN Number based Filtering (SNF), enables edge routers of the ISP network to validate incoming SYN-ACK packets destined to the victim without maintaining state information about *individual* SYN packets sent originally by the victim. The main idea of the SNF scheme is to restrict the choice of the Initial Sequence Numbers (ISNs) of SYN packets generated by the victim to certain pattern, such that corresponding SYN-ACK packets can be validated at the ISP perimeter. In the following subsections, we develop two variants of the SNF scheme to counter both *classical* and *advanced* TCP-based RDoS attacks. Classical attacks are those in which the attacker does not react to defenses employed by the victim during attack period. Advanced attacks are those in which the attacker monitors victim's reaction and acts accordingly. In the rest of this section, we develop a simple analytical models to evaluate the performance of the proposed scheme under both classical and advanced attacks focusing on the following performance metrics:

- False positive rate (FPR): The percentage of attack SYN-ACK packets that are falsely allowed to pass the ISP perimeter toward the victim.
- False negative rate (FNR): The percentage of legitimate SYN-ACK packets that are falsely filtered.

Clearly, we need both metrics to be minimized.

4.1 Countering Classical TCP-Based RDoS Attacks

In the classical TCP-based RDoS attacks, SYN packets generated by attack nodes (recall that TCP-based reflector attacks start by generating spoofed SYN packets that hit the reflectors, forcing them to flood the victim with the corresponding SYN-ACK packets) will continue to hold ISNs according to the rules specified originally by the attack tool itself. To counter such attacks, we propose the Basic-SNF scheme in which the victim chooses a *secret pattern* for its ISNs. The secret pattern, C_s, is set by fixing a randomly chosen k-bits out of the 32 sequence number bits to form a specific combination. The victim then initiates a filtering request by multicasting a message to all edge routers of the ISP network. The message includes the specific combination, C_s, that should be used to validate incoming SYN-ACK packets. Edge routers inspect incoming SYN-ACK packets destined to the victim according to the algorithm shown in figure 2. In this algorithm, if the (ACK number - 1) of the SYN-ACK packet contains the secret pattern, C_s, then the packet is passed. Otherwise, it is filtered.

```
Basic-SNF (SYN-ACK packet $P$)
  a. $X = P$.ACK_number-1
  b. if ($X$ contains $C_s$) pass P
  c. else drop P
```

Fig. 2. Basic-SNF algorithm. This algorithm is performed at each edge router while attack is going on. It is applied only to SYN-ACK packets destined to the victim

Assuming that the ISNs of attack packets are generated randomly, the probability that a given attack packet will contain a bit combination that matches the secret pattern currently in use, C_s, is equal to $\frac{1}{2^k}$, where k is the length of C_s. This implies that the Basic-SNF scheme is very effective against classical attacks. However, a major weakness of the this scheme is that the attacker does not have to discover the actual secret pattern used by the victim. In fact, it is sufficient to intercept a single legitimate SYN or SYN-ACK packet and using its corresponding ISN as an input to subsequent attack packets. In order to address this issue, we propose to change the secret pattern, C_s, in a way that significantly reduces attacker's chances of launching replay attacks (i.e., using previously used secret pattern). To achieve this, the secret pattern can be changed either periodically; where the C_s is changed regularly every T time units, or reactively; where C_s is changed whenever a replay attack is detected by the victim.

4.2 Countering Advanced TCP-Based RDoS Attacks

In this model, we assume that an attacker experiences some delay D_a before being able to reconfigure its attack tool to generate SYN packets that carry valid SYN-numbers to the reflectors (i.e., packets that hold the secret pattern currently in use by the victim). This delay consists of the time required to intercept SYN packets generated by the victim itself, the communication time to the zombies under attacker's control, and the attack tool reconfiguration time. To deal with this type of attacks, we modify the Basic-SNF scheme by changing the secret pattern periodically.

Analysis of Periodic-SNF. In this scheme, we propose changing the secret pattern of the SYN number periodically (i.e., every T time units). This is done by dividing the time since attack is detected into fixed intervals, each of length T. A secret pattern, C_i, is then assigned for the i-th interval, for $i = 1, 2, 3, ...$ It is assumed that interval length and secret pattern assignment are known to edge routers via authentic multicasting. Any SYN packet generated by the victim during the i-th interval must hold C_i as part of its ISN. For SYN-ACK packet validation, each edge router acts independently according to the filtering algorithm shown in Fig. 3.

The edge router specifies the interval in which the packet to be inspected has arrived (step a). It is expected that SYN-ACK packets that correspond to SYN packets which are sent at the end of a given interval (e.g., the $(i-1)$-th interval) may arrive at the edge router during the subsequent interval (i.e., the i-th interval). Such packets will be dropped if the validation is done based on the current secret pattern in use (i.e., C_i) alone. To reduce the impact of this problem, a packet is considered to be valid if its secret pattern matches either of C_i or C_{i-1} given that its arrival time falls between $(i-1)T$ and $(i-1+\alpha)T$, where $0 \leq \alpha \leq 1$ (step c). Otherwise the validation is performed based on C_i alone (step d).

Fig. 4 shows two scenarios for connection establishment by the victim assuming α to be zero. In scenario 1, the request (i.e., a SYN packet) is sent by the victim at time t_1 during the $(i-1)$-th interval, the reply (i.e., the corresponding SYN-ACK packet) takes x_1 time units to arrive at the ISP perimeter. Since $(t_1 + x_1 < T)$, the reply is passed. In scenario 2, the reply that corresponds to the request made at t_2 arrives the ISP perimeter at $t_2 + x_2 > T$ (i.e., in the i-th interval. The reply is filtered in this case because it holds a secret pattern (C_{i-1}) that is no longer in use.

```
Periodic-SNF (SYN-ACK packet $P$)
  a. $i = \lceil \frac{P.at}{T} \rceil$
  b. $X$=$P$.ACK-1
  c. if ( $(i-1)T \leq P.at \leq (i-1+\alpha)T$)
       • if (($X$ contains $C_i$) OR ($X$ contains $C_{i-1}$)) pass P
  d. else if ($X$ contains $C_i$) pass P
  e. else drop P
```

Fig. 3. Periodic-SNF algorithm. This algorithm is performed at each edge router while attack is going on. It is applied only to SYN-ACK packets destined to the victim. *P.at* stands for the packet arrival time. P. C_i stands for the secret pattern assigned for interval i. α is the overlapping ratio

Generally, because of the attacker's ability to generate valid attack packets after D_a seconds each time a new secret pattern is applied (recall that D_a represents the amount of delay experienced by the attacker before being able to

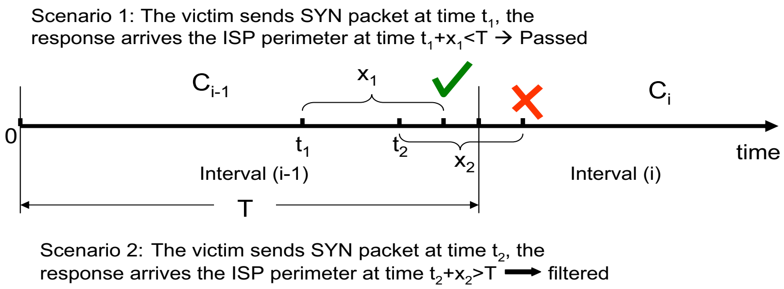

Scenario 1: The victim sends SYN packet at time t1, the response arrives the ISP perimeter at time t1+x1<T → Passed
C i-1
x1
✓
✗
C i
0
t1
t2
x2
time
Interval (i-1)
Interval (i)
T
Scenario 2: The victim sends SYN packet at time t2, the response arrives the ISP perimeter at time t2+x2>T → filtered

Fig. 4. The impact of changing the secret pattern of the ISN for connections generated by the victim. The secret pattern is set to C_{i-1} during the $(i-1)$-th interval, and to C_i during the i-th interval

reconfigure its tool), it can be seen that the attacker can pass its packets during the window of time specified by $(1+\alpha)T - D_a$. It is clear that the choice of T and α introduces a tradeoff that shapes the false positive and false negative rates. The following analysis focuses on evaluating both metrics.

FPR for Periodic-SNF ($FPR_{periodic}$): $FPR_{periodic}$ can be expressed as the expected value of the ratio of time in which the attack traffic penetrates the ISP perimeter. Let $A_{periodic}$ represent this ratio. It is clear that:

$$A_{periodic} = \frac{(1+\alpha)T - D_a}{(1+\alpha)T} \tag{1}$$

Assuming that D_a has an exponential distribution[2] with rate μ_a, the expected value of $A_{periodic}$ is given by Equation (2).

$$FPR_{periodic} = E[A_{periodic}] = (1+\alpha) - \frac{1}{\mu_a(1+\alpha)T} \tag{2}$$

FNR for Periodic-SNF ($FNR_{periodic}$): In order to gain an insight on the performance of the periodic *SNF* scheme in terms of the false negative rate $FNR_{periodic}$, we assume that the victim is generating TCP connections at a Poisson rate of λ per unit time, and that the distribution of connection response time X (defined as the time between sending the SYN packet by the victim and receiving the corresponding SYN-ACK at the perimeter) is given by $X = c + Y$, where c is a constant and Y is a variable component. This assumption is based on the definition of the round trip time (RTT), which consists of a *fixed* component given by the sum of link propagation latencies and transmission and processing delays on all nodes in the forward and reverse direction, and additional *variable* components due to queuing and processing delays at overloaded routers and end-hosts. Assuming that Y can be expressed as the summation of n identical exponentially distributed random variables each with rate β, Y is said to be an Erlang random variable with parameters β and n.

[2] This assumption is made because actual measurements of D_a are not available.

Based on these assumptions, we are interested in finding an expression for $FNR_{periodic}$, which is the same as the probability of filtering a legitimate packet. The following equation represents the probability distribution function of X:

$$f_X(x) = \begin{cases} \frac{\beta^n (x-c)^{n-1} e^{-\beta(x-c)}}{(n-1)!} & \text{if } x > c; \\ 0 & \text{if } x \leq c. \end{cases}$$

We proceed in our analysis for $x > c$. The cumulative distribution function (CDF) of X is given by:

$$F_X(x) = 1 - \sum_{k=0}^{n-1} \frac{e^{-\beta(x-c)} \beta^k (x-c)^k}{k!} \tag{3}$$

It can be seen that Pr(a legitimate packet P is filtered | P is sent at time t) = Pr($X \geq (1+\alpha)T - t$ | P is sent at time t) = 1 - Pr($X \leq (1+\alpha)T - t$ | P is sent at time t). We call this probability the conditional probability of legitimate packet filtering $CPLP_{periodic}$. This probability is given by the following equation:

$$CPLP_{periodic} = 1 - F_X((1+\alpha)T - t) \tag{4}$$

Substituting 3 with ($x = (1+\alpha)T - t$) in 4, we obtain:

$$CPLP_{periodic} = \sum_{k=0}^{n-1} \frac{e^{-\beta((1+\alpha)T-t-c)} \beta^k ((1+\alpha)T - t - c)^k}{k!} \tag{5}$$

Unconditioning on the time at which each request is generated, we obtain:

$$FNR_{periodic} = \frac{1}{T} \int_0^T CPLP_{periodic} dt \tag{6}$$

In practice, each end-to-end path is characterized by its own minimum possible RTT (i.e., the fixed component c of the RTT along the given path). Also, the variable component represented by the erlangian variable Y has its own parameters β and n for each TCP segment due to the diversity of destinations and the variability of load on network routers. However, for the purpose of obtaining an upper bound on the probability of a legitimate SYN-ACK packet being filtered, due to secret pattern change, we can assume worst case values for c, β, and n.

We performed extensive simulation experiments to evaluate the value of $FNR_{periodic}$ for several values of α and T. It is to be noted that $FNR_{periodic}$ does not depend on λ since all arrivals are independent. However, the value of λ was fixed to 0.004 just for the sake of experimentation. Fig.5 plots the percentage of false positive and false negative rates side by side to gain an insight on the tradeoff introduced by α and T. To obtain a worst case value for the random variable X, we assigned a value of 1 second to the constant c, which was found to be the maximum reported minimum RTT [1]. Also, to eliminate the variability of parameters n and β, we assigned the values 30 and 2 for them,

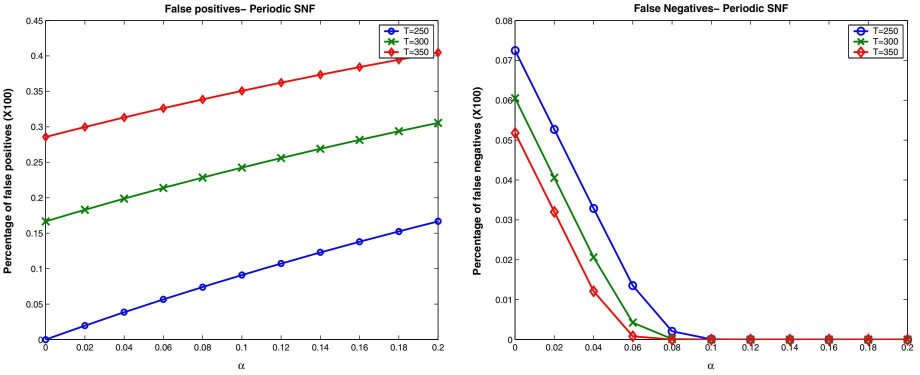


Fig. 5. The effect of α on the (left) percentage of false positive and (right) percentage of false negative for different values of T. μ_a, λ, β, and n were fixed to 0.004, 5, 2, and 30, respectively

respectively. This corresponds to the maximum reported path length [4], and a relatively large queuing and processing delay at intermediate routers. α was varied along the x axis in the range of 0 to 0.2.

By recalling that αT represents the amount of extra time in which a secret pattern of a given interval remains valid throughout the subsequent interval, it is easy to explain the decrease of the false negative rate by increasing α. It is also straightforward to explain the increase of false positive rate at the same time. It can be observed that as T increases, the false negative rate decreases. This is expected since secret pattern change will be less frequent, resulting in lower percentage of legitimate packets being dropped. At the same time, the false positive rate increases sharply. For example, the false positive rate exceeds 30% for $T = 350$. This is due to the fact that increasing T provides larger window of time for the attacker to pass his traffic toward the victim. These results lead to an important conclusion: The Periodic-SNF scheme is expected to perform well in terms of false negative rate. However, it is expected to perform well in terms of false positive rate only if the value of T happens to be close to the rate at which the attacker reconfigures its tool with a valid secret pattern. As a future work, we plan to investigate the effectiveness of a reactive version of the SNF scheme in which the secret pattern change is triggered by certain event rather changing it periodically.

4.3 Practical Consideration

In this section, we discuss the practicality of the proposed schemes focusing on the following two aspects.

The Impact of Restricting the ISN to Specific Pattern: It is known that when an end system sends its SYN packet to establish a TCP connection, it chooses an (ISN) for that connection. Typically, the ISN should change over time, so that each connection has a different ISN. New instances of a connection may be established if the connection is opened and closed in quick succession, or if the connection breaks with loss of memory and is then reestablished. The TCP should be able to identify duplicate segments from previous instances of the same connection. To achieve this, RFC 793 [16] specifies that the ISN should be viewed as a 32-bit counter that is incremented by one every 4 microseconds. Although this practice is common in most TCP implementations, it is not necessary for proper functionality of the protocol, and it can be violated in favor of mitigating the effect of an ongoing attack. The main impact of restricting the choice of ISN is the increase the possibility of *duplicate segment problem* (i.e., having duplicate segments from previous instances of the same connection) if multiple instances of the same connection are created frequently. However, this drawback can be tolerated during TCP-based RDoS attack, because it is more important to mitigate the effect of the attack. It is also, important to point out that restricting the ISN to certain pattern does not increase the vulnerability of connection hijacking through TCP sequence number guising. This argument is based on the fact that for a connection to be hijacked, the sequence number of the server, rather the client, needs to be guessed.

The Need to Perform TCP's Header Inspection: The proposed scheme requires that all edge routers of the ISP network that contains the victim to inspect each packet closely enough to determine if it is a TCP SYN-ACK packet, so the mechanism can more closely analyze the packet to see if it is a legitimate SYN-ACK. This requires, at least for all TCP packets destined to the victim, examining the TCP header fields. Fortunately, recent advancements in router manufacturing technology has allowed performing such functions at high Internet speeds. For example, Cisco routers currently have the TCP intercept feature [6] which implements a software to protect TCP servers from TCP SYN-flooding attacks. In this context, a router is configured to perform TCP intercept in which the software actively intercepts each incoming connection request (SYN) for the purpose of legitimacy check. We believe that a similar approach can be adopted to support the proposed SNF scheme. The main difference will be in intercepting SYN-ACK packets instead of SYN packets, and accordingly the router should be configured to perform packet filtering based on the specified rules.

5 Conclusions

TCP-based reflector distributed denial of service attacks represent a challenging problem. Using SYN-ACK packets to flood certain system complicates the process of distinguishing these packets from legitimate packets destined to the same system. In this paper, we developed the concept of victim-assistance to mitigate such attacks. Then, we proposed a scheme, called SYN number-based filtering (SNF), which is based on the idea of restricting victim's choice of the initial sequence numbers of its generated connection establishment requests during an attack, such that legitimate incoming SYN-ACK packets can be verified at the ISP perimeter by checking specific bit pattern.

Through analytical studies, we showed that the two variants of the SNF scheme (i.e., the Basic-SNF and the Periodic-SNF) offer performs very well for classical attacks and advanced attacks, respectively. Our analysis shows that the Basic-SNF scheme is very effective against classical attacks as it ensures an extremely low false positive rate and exactly zero false negative rate. For advanced attacks, the periodic-SNF performs very well in terms of false positive and false negative rates especially if the period of changing the secret pattern is close to attacker's response time.

Obtaining victim's assistance represents the basis in defending against TCP -based RDoS attacks in the proposed scheme. The importance of this approach is reflected in the fact that new source of information (i.e., the victim) is now available to make distinction between attack and legitimate packets. This approach opens new directions of research in designing efficient defenses for DoS attacks, which includes (1) designing and analyzing a reactive version of the SNF scheme to counter advanced DoS attacks more efficiently, and (2) investigating the viability of victim's assistance in defending against other types of DoS attacks. In this context, several issues needs to be addressed. For example, what information should be provided by the victim? How this information is to be communicated to edge routers? When to activate/terminate the mitigation scheme and on which edge routers? How to extend the concept of victim-assistance to inter-domain setting?

Acknowledgments

The authors would like to thank Prof. Ahmed E. Kamal for his constructive suggestions and help to improve the quality of the work presented in this paper.

References

1. J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in TCP Round-trip Times," in *Proc. of the ACM SIGCOMM Internet Measurement Conference (IMC'03)*, Miami, FL, October 2003.
2. S. Chen and Q. Song, "Perimeter-Based Defense against High Bandwidth DDoS Attacks," to appear in *Proc. IEEE Transactions on Parallel and Distributed Systems (TPDS 2005).*
3. CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks. Available at: http://www.cert.org/advisories/CA-1998-01.html.
4. B. Cheswick, H. Burch, and S. Branigan, "Mapping and Visualizing the Internet", in *Proc. USENIX Annual Technical Conference 2000.*
5. Cisco systems, "Defeating DDoS Attacks," white paper.
6. Cisco systems, "Configuring TCP Intercept (Prevent Denial-of-Service Attacks)," available at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt3/scdenial.htm

7. P. Ferguson and D. Senie, “Network ingress filtering: Defeating denial-of-service attacks which employ IP source address spoofing,” RFC 2827, 2000.
8. S. Gibson,“Distributed Reflection Denial of Service,” February 22_{nd}, 2002. Available at http://grc.com/dos/drdos.htm.
9. C. Jin, H. Wang, and Kang G. Shin,“ Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic,” in *Proc. ACM Conference on Computer and Communications Security (CCS)'2003*, Washington, DC, October 2003.
10. A. Hussain, J. Heidemann, and C. Papadopoulos,“A Framework for Classifying Denial of Service Attacks,” in *Proceedings of ACM SIGCOMM 2003.*
11. J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang., “SAVE: Source address validity enforcement protocol,” In *Proc. of IEEE INFOCOMM 2002*, April, 2002.
12. M. Mellia, I. Stoica, and H. Zhang, “TCP Model for Short Lived Flows,” in *Proc. IEEE Communication Letters*, Feb. 2002.
13. V. Paxson, “An analysis of using reflectors for distributed denial-of-service attacks,” in *Proc. Computer Communication Review 31(3)*, July 2001
14. T. Peng, C. Leckie and K. Ramamohanarao, “Protection from Distributed Denial of Service Attack Using History-based IP Filtering,” in *Proc. of IEEE International Conference on Communications (ICC2003),* May 2003.
15. T. Peng, C. Leckie and R. Kotagiri,“Detecting reflector attacks by sharing beliefs,” in *Proc. IEEE Global Communications Conference*, October, 2003
16. J. Postel, “Internet Protocol - DARPA Internet Program Protocol Specification,” RFC 791.

Performance Analysis of the Uplink of a CDMA Cell Supporting Elastic Services

Gábor Fodor[1] and Miklós Telek[2]

[1] Ericsson Research, SE-164 80 Stockholm, Sweden
Gabor.Fodor@ericsson.com
[2] Budapest University of Technology and Economics,
H-1111 Budapest, Hungary
telek@hit.bme.hu

Abstract. We consider a single cell of a multi-service CDMA network, in which some of the service classes are explicit rate controlled. We call these elastic service classes. The instantaneous bit rate of elastic services is dynamically adjusted between a minimum and maximum value such that the system always remains work conserving. We develop a Markov model that allows us to study the impact of such state dependent (dynamic) rate control on the class-wise blocking probabilities and the first two moments of the holding times. We conclude that dynamic (state dependent) rate adjustment decreases the class-wise blocking probabilities and only moderately increases the expectation and the second moment of the time spent in the system.

1 Introduction

The Erlang capacity of code division multiple access (CDMA) systems has long been in the focus of research, basically starting ever since code division was first proposed as a multiple access technique; see for instance Chapter 6 of [1] and Section 5.7 of [2] which are dedicated to Erlang capacity issues in CDMA networks. Recently, the seminal paper by Altman [3], (see also [4]) showed that assuming perfect power control, slowing the transmission rates in the case of a single cell with heterogeneous QoS characteristics increases capacity. This result is non-trivial, since slowing the transmission rate leads to increased holding time of the session. Specifically, consider a multi-service system at which sessions arrive according to a Poisson process with intensity λ and assume that the session holding times are exponentially distributed with parameter μ. Then, assuming a *fixed* slow down rate a, the load (and thereby the Erlang capacity) of the multi-service system can increase from $\rho = \sum_i \lambda_i/\mu_i$ to $\rho_a > \rho$ ($a > 1$) under the same class-wise blocking probability constraints. The assumption on the system state independent fixed slow down rate a leads to a product form solution for the steady state that allows for a straightforward derivation of the blocking probabilities.

However, slowing down the transmission rates in all system states leads to a non-optimal resource utilization. Therefore, in this paper we propose a model that allows the instantaneous bit-rate of the elastic service classes to *adaptively* decrease/increase depending on the system state. The relaxation of the assumption that a is fixed is motivated

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 205–216, 2005.

Table 1. Model (Input) Parameters

K	Number of service classes
$R_p(k)$	Peak bit rate associated with class-k sessions
$\lambda(k)$	Arrival intensity of sessions belonging to class-k
$1/\mu(k)$	Mean (nominal) holding time of sessions belonging to class-k
$\hat{a}(k)$	Maximum slow down (using the terminology of [3]) of $R_p(k)$
i	Other cell (sector) interference to own cell (sector) interference ratio
$E(k)/N_0$	Normalized signal energy per bit requirement of class-k

by the observation that assuming perfect power control, the individual session bit-rates may change dynamically in the system depending on the currently on-going number of sessions and their service class requirements. We intuitively expect that slowing down the session rates only in those system states in which it is necessary (as opposed to slowing down the elastic sessions in *all* system states) will lead to lower blocking probabilities and higher resource utilization. As we shall see, this is indeed the case. Unfortunately our setting destroys the product form solution but leads to a quasi birth-death structure that still allows for efficient solution of the system, at least when the number of service classes is low.

It has been observed by several papers that for elastic services not only the first, but also the second moment of the actual holding time has an impact on the user perceived QoS [5], [6]. Therefore, we also develop a modified Markov model that allows us to arrive at numerical results for the second moment of the elastic class holding time.

The paper is structured as follows. In the next Section we formulate the model and state the analysis objective in terms of performance measures of interest. Next, Section 3 examines the system state space and develops a solution for the steady state, from which various performance measures are derived. Subsequently, in Section 4, we describe the elements and the solution of the Markov model that allows us to derive the higher moments of the time spent in the system by elastic flows. Section 6 draws conclusions.

2 Model Assumptions and Performance Analysis Objective

2.1 The Input Parameters

We consider a single CDMA cell at which sessions belonging to one of K service classes arrive according to a Poisson arrival process of intensity $\lambda(k)$ ($k = 1, \ldots, K$). Each class is characterized by a peak bit-rate requirement $R_p(k)$ and an exponentially distributed nominal holding time with parameter $\mu(k)$. When sending with the peak rate for a session, the required target ratio of the received power from the mobile terminal to the total interference energy at the base station is calculated as follows:

$$\tilde{\Delta}(k) = \frac{E(k)}{WN_0} \cdot R_p(k), \quad k = 1, \ldots, K,$$

where $E(k)/N_0$ is the signal energy per bit divided by the noise spectral density that is required to meet a predefined QoS (e.g. bit error rate, BER); noise includes both thermal noise and interference. This required $E(k)/N_0$ can be derived from link level simulations and from measurements. $R_p(k)$ is the peak bit rate of the session of class k and W is the spread spectrum bandwidth.

Just like in [3], let $M(k)$ be the number of ongoing sessions of class k. We will refer to vector $\boldsymbol{M} = \{M(k)\}, \quad k = 1, \ldots, K$ as the *state* of the system. We now assume that arriving sessions are blocked by a suitable admission control algorithm that prevents the system to reach the state in which the power that should be received at the base station would go to infinity. In other words, a suitable admission control algorithm must prevent the system to reach its *pole capacity* (as defined by Equation (8.10) of [7] and (5) of [3]).

In order to model this, we use the standard equations (8.3)-(8.12) from [7] as follows.

The power $P(k)$ that is received at the base station from the mobile terminal for session k must fulfill (assuming that the terminal can control the power level for each session separately):

$$\frac{P(k)}{P_N + I_{own} + I_{other} - P(k)} = \tilde{\Delta}(k), \quad k = 1, \ldots, K \tag{1}$$

where I_{own} is the total power received by the base station within its cell (or sector), and I_{other} is the total power received from other cells (or sector). P_N is the background noise power. That is, $I_{own} = \sum_{k=1}^{K} M(k)P(k) \quad \text{and} \quad I_{other} = i \cdot I_{own}$.

Rewriting (1), we get:

$$\frac{P(k)}{P_N + I_{own} + I_{other}} = \Delta(k), \quad \Delta(k) = \frac{\tilde{\Delta}(k)}{1 + \tilde{\Delta}(k)}, \quad k = 1, \ldots, K \tag{2}$$

From (2):

$$\frac{P(k)}{P_N + (1+i)\sum_{l=1}^{K} M(l)P(l)} = \Delta(k), \quad k = 1, \ldots, K$$

Multiplying with $M(k)$ and summing over k:

$$\sum_{k=1}^{K} P(k)M(k) = \frac{P_N \sum_{l=1}^{K} M(l)\Delta(l)}{1 - (1+i)\sum_{l=1}^{K} M(l)\Delta(l)} \tag{3}$$

From (3):

$$P(k) = \left(P_N + (1+i)\sum_{l=1}^{K} M(l)P(l) \right) \cdot \Delta(k), \quad k = 1, \ldots, K \tag{4}$$

Substituting (3) into (4) and denoting $\varOmega = \sum_{l=1}^{K} M(l) \cdot \varDelta(l)$:

$$P(k) = \left(P_N + (1+i) \cdot \frac{P_N \cdot \varOmega}{1-(1+i)\cdot \varOmega} \right) \cdot \varDelta(k) = \frac{P_N \cdot \varDelta(k)}{1-(1+i)\cdot\varOmega}, \quad k = 1, \ldots, K \tag{5}$$

From (5) it is clear that the admission control algorithm must prevent that $\varOmega$ reaches $1/(1+i)$. In the single class case, it means that the number of admitted sessions must fulfill: $M < \lfloor \varOmega/\varDelta \rfloor$ (where we now let $M = M(1)$ and $\varDelta = \varDelta(1)$). In the rest of the paper we assume that such admission control algorithm operates in the system. We note that because of the relation expressed by Equation (2) and the definition of $\varOmega$, one can consider $\varOmega$ as the overall resource in the multi-rate CDMA system that the sessions must share. This observation can be seen as an analogy between the multi-rate CDMA model and the multi-rate loss models developed in the 80's and 90's [8]. The major difference between the classical loss models and the present CDMA model is that the relation between the slow down rate $a(k)$ and the resource consumption $\varDelta_a(k)$ is not linear, as we shall see it in the next subsection and as it is also reflected by Equation (7).

2.2 The Impact of Slow Down

Recall that the required target ratio ($\varDelta(k)$) depends on the required bit-rate. Explicit rate controlled elastic services tolerate a certain slow down of their peak bit-rate ($R_p(k)$) as long as the actual instantaneous bit rate remains greater than the minimum required $R_p(k)/\hat{a}(k)$. When the bit rate of a class-k session is slowed down to $R_p(k)/a(k)$, ($0 < a(k) \leq \hat{a}(k)$) its required $\varDelta_a(k)$ value becomes:

$$\varDelta_a(k) = \frac{\tilde{\varDelta}(k)}{a(k) + \tilde{\varDelta}(k)} = \frac{\varDelta(k)}{a(k)\cdot(1-\varDelta(k)) + \varDelta(k)}, \quad k = 1, \ldots, K \tag{6}$$

which increases the number of sessions that can be admitted into the system, since now $\varOmega_a$ must be kept below $1/(1+i)$, where

$$\varOmega_a = \sum_{k=1}^{K} M(k) \cdot \varDelta_a(k).$$

We use the notation $\varDelta_{min}(k) = \varDelta_{\hat{a}}(k)$ to denote the class-wise minimum target ratios (can be seen as the minimum resource requirement), that is when the session bit-rates of class-k are slowed down to that class' minimum value. It is the task of the bandwidth sharing policy to determine the $\varDelta_a(k) \geq \varDelta_{min}(k)$ values (and consequently the $a(k) \leq \hat{a}(k)$ class-wise instantaneous slow down factors) for each state of the system such that $\varOmega_a < 1/(1+i)$. Because of the admission control assumption, such a resource assignment is always possible in feasible states.

2.3 Performance Measures

In order to define the performance measures of interest in this system, we make the following considerations. It is intuitively clear that the residency time of the elastic flows in this system depends not only on the amount of data they want to transmit (which is a random variable), but also on the instantaneous bit-rate they receive during their holding times. In order to specify this relationship we define the following quantities:

- $\theta(t,k) = R_a(t)$ defines the *instantaneous throughput* of a class-k elastic flow at time t, Note that $\theta(t,k)$ is a discrete random variable for any $t \geq 0$.
- $T_x(k) = \inf\{t \mid \int_0^t \theta(\tau,k)d\tau \geq x\}$ (random variable) gives the time it takes for the system to transmit x amount of data through a class-k elastic flow,
- $\hat{\theta}_x(k) = x/T_x(k)$ defines the *conditional throughput* of the class-k elastic flow during the transmission of x data unit. Note that $\theta_x(k)$ is a continuous random variable.
- $\hat{\theta}(k) = \int_0^\infty \hat{\theta}_x(k)\ dG(x) = \mu(k)/R_p(k) \int_0^\infty \hat{\theta}_x(k)\ e^{-x\ \mu(k)/R_p(k)}\ dx$ (random variable) defines the *throughput* of the class-k elastic flow, where the amount of transmitted data is exponentially distributed with parameter $\mu(k)/R_p(k)$.

We note that from a user's perspective, it is in fact not the throughput, but the time it takes to transfer a file (image, email, etc) that has an impact on the perceived QoS. Therefore, in the numerical section we will focus on this performance measure.

In addition, we are interested in finding the blocking probabilities for all service classes i.e., $B(k), k = 1, \ldots, K$.

The instantaneous and average *utilization* of the system are defined as follows:

$$U(t) = \sum_{k=1}^{K} M(t,k) \cdot \Delta_{a(t)}(k); \quad E[U(t)] = \int_0^\infty \tau\ dF_U(\tau),$$

where $F_U(\tau)$ denotes the stationary distribution of $U(t)$. In the above definitions it is emphasized that both $M(k)$ and $a(k)$ evolve in time.

3 Determining the Steady State of the System

3.1 Determining the System State Space

The maximum number of sessions from each class can be calculated as follows:

$$M_{max}(k) = \lfloor (\Delta_{\hat{a}(k)})^{-1} \rfloor, \quad k = 1, \ldots, K$$

Recall that in each $\boldsymbol{M}$ state of the system, the inequality $\sum_k M(k) \cdot \Delta_a(k) < 1/(1+i)$ must hold. The states that satisfies this inequality are the *feasible states* and constitute the state space of the system which we will denote by Γ. The feasible states, in which the acceptance of an additional class-k session would result in a state outside of the state space are the class-k *blocking states*. The set of the blocking states is denoted by Γ^B. Due to the "Poisson Arrivals See Time Averages" (PASTA) property, the sum of the class-k blocking state probabilities gives the (overall) class-k blocking probability [8].

In each feasible state, it is the task of the bandwidth sharing policy to determine the $\Delta_a(k) = f(\boldsymbol{M})$ values for each class. From these, the class-wise slow down factors and the instantaneous bit-rates of the individual sessions can be calculated as follows:

$$a(k) = \frac{\Delta(k)\cdot(1-\Delta_a(k))}{\Delta_a(k)\cdot(1-\Delta(k))}; \quad R_a(k) = R_p(k)/a(k) \tag{7}$$

3.2 Determining the Generator Matrix

For ease of presentation, but without loosing generality, we use the example illustrated by Figure 1 to develop the generator matrix of the state space. Assume that $i = 0$, $\hat{a}(1) = 1$, $\hat{a}(2) > 1$ and $\hat{a}(3) > 1$. In this case, the task of the bandwidth sharing policy simplifies to determining $\Delta_a(2)$ and $\Delta_a(3)$ for each state, from which $\hat{a}(2)$ and $\hat{a}(3)$ follows immediately.

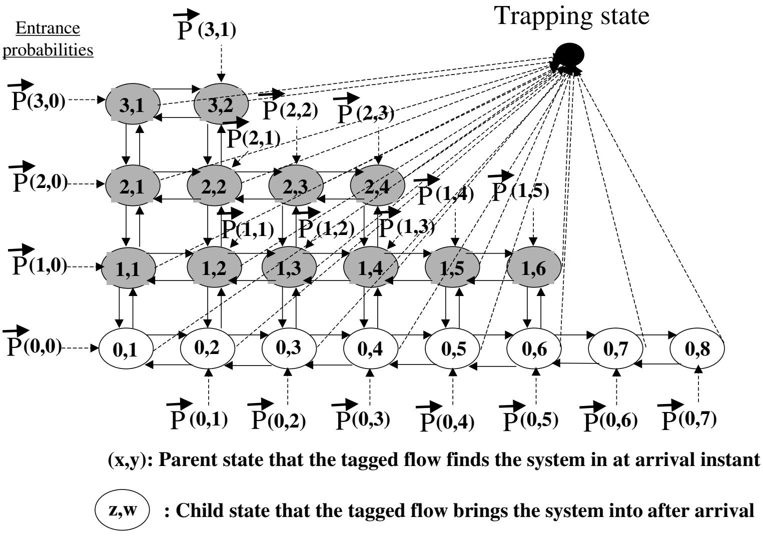


Fig. 1. Modified State Space with Trapping State

We now make use of the assumptions that the arrival processes is Poisson and the nominal holding times are exponential. The system under these assumptions is a continuous time Markov chain (CTMC) whose state is uniquely characterized by the state vector $\boldsymbol{M}$. In order to arrive at the performance measures of interest, we need to determine the CTMC's generator matrix $\mathbf{Q}$ and its steady state solution $\boldsymbol{P}$.

Based on the considerations of the preceding subsections, we see that the generator matrix $\mathbf{Q}$ possesses a nice structure, because only transitions between "neighboring states" are allowed in the following sense. Let $q(m_1, m_2, m_3 \rightarrow m_1', m_2', m_3')$ denote the transition rate from state (m_1, m_2, m_3) to state (m_1', m_2', m_3'). Then the non-zero transition rates between the feasible states are:

$$\begin{aligned}
q(m_1, m_2, m_3 \to m_1 + 1, m_2, m_3) &= \lambda_1 \\
q(m_1, m_2, m_3 \to m_1, m_2 + 1, m_3) &= \lambda_2 \\
q(m_1, m_2, m_3 \to m_1, m_2, m_3 + 1) &= \lambda_3 \\
q(m_1, m_2, m_3 \to m_1 - 1, m_2, m_3) &= m_1 \cdot \mu_1 \\
q(m_1, m_2, m_3 \to m_1, m_2 - 1, m_3) &= m_2 \cdot \mu_2 / a_2 \\
q(m_1, m_2, m_3 \to m_1, m_2, m_3 - 1) &= m_3 \cdot \mu_3 / a_3
\end{aligned}$$

The first three equations represent the state transitions due to session arrivals, while the second three equations represent the transitions due to session departures. Here we utilized the fact that class-1 sessions cannot be slowed down, while class-2 and class-3 sessions can be slowed with a maximum of $\hat{a}_2$ and $\hat{a}_3$ factor respectively.

Note that the derivation of the generator matrix relies on the fact that the system is Markovian. This is not trivial because one could intuitively argue that since the elastic flows bring with themselves a certain amount of workload, the memoryless property does not hold, even if this workload is exponentially distributed. However, the Markovian property for such systems was independently of one another observed and formally proven by Altman *et al.* [9], Andersen *et al.* [5] and Nunez Queija *et al.* [10].

3.3 Deriving the Performance Measures

With the generator matrix in hand, we now need to solve the equation $\boldsymbol{P} \cdot \mathbf{Q} = 0$ (taking into account that $\boldsymbol{P} \cdot \boldsymbol{e} = 1$, where $\boldsymbol{e} = (1, \ldots, 1)$). From the steady state distribution $\boldsymbol{P}(\boldsymbol{M})$, the performance measures of interest (except the second moments) immediately follow.

The blocking probabilities and the mean number of class-k sessions in the system are straightforward to derive (recall that $M(k) = \boldsymbol{M}[k]$) :

$$B(k) = \sum_{\boldsymbol{M} \in \Gamma^B} \boldsymbol{P}(\boldsymbol{M}), \quad E[M(k)] = \sum_{\boldsymbol{M} \in \Gamma} M(k) \cdot \boldsymbol{P}(\boldsymbol{M}).$$

From this and Little's theorem we obtain the mean time that a class-k session spends in the system and the average resource utilization:

$$T(k) = \frac{E[M(k)]}{\lambda(k) \cdot (1 - B(k))}, \quad E[U] = \sum_{\boldsymbol{M} \in \Gamma} \sum_{k=1}^{K} M(k) \cdot \Delta_a(k).$$

From the steady state, the average class-wise throughput also follows:

$$\hat{\theta}(k) = \frac{\sum_{\boldsymbol{M} \in \Gamma} M(k) \cdot \boldsymbol{P}(\boldsymbol{M}) \cdot (1/a(\boldsymbol{M}))}{\sum_{\boldsymbol{M}} M(k) \cdot \boldsymbol{P} \cdot (\boldsymbol{M})}, \tag{8}$$

which provides an easy way to check and verify results, since the normalized throughput (to $R_p(k)$) is the reciprocal of the normalized mean holding time (to $1/\mu(k)$), which is indeed the case as we will see in the numerical section.

4 Solution Approach Based on a Markov Model with a Trapping State

The higher moments of the time spent in the system by the elastic sessions requires additional effort, which is the topic of this section.

4.1 Session Tagging and Modifying the State Space

We will continue to think of an elastic session as one that brings with itself an exponentially distributed amount of work and, if admitted into the system, stays in the system until this amount of work is completed. The method we follow here is based on (1) *tagging* an elastic session arriving to the system, which, at the time of arrival is in one of the feasible states; and (2) carefully examining the possible transitions from the moment this tagged call enters the system until it acquires the required service and leaves the system. Finally, un-conditioning on all possible entrance state probabilities, and applying results from [12], the moments of the best effort service time can be determined.

In this modified state space, we also define a *trapping* (*absorbing*) *state*, which corresponds to the state where the tagged session has acquired the requested amount of service and leaves the system. In this expanded state space the time until absorption [12] corresponds to the time the tagged session spends in the system. Indexing the modified state space in a similar manner as the original state space in Section 3, the new generator matrix, $\tilde{\mathbf{Q}}$, will have the following structure:

$$\tilde{\mathbf{Q}} = \begin{bmatrix} B & T \\ 0 & w \end{bmatrix} \tag{9}$$

where the B matrix represents the transitions between the non-trapping states, the T vector contains the transitions *to* the trapping state, the 0 vector indicates that no transitions are allowed *from* the trapping state, and $w = 0$. Once the structure of the expanded state space and the associated transition rates together with the initial probability vector, $P_R(0)$, are determined, we can apply the result of [13] (proved also in [14]) for the determination of the r^{th} moment of T_x:

$$T^{(r)} = r! \cdot P_R^T(0) \cdot (-B)^{-r} \cdot e \tag{10}$$

and specifically for the mean:

$$E[T] = P_R^T(0) \cdot (-B)^{-1} \cdot e, \tag{11}$$

which provides a natural way to check the results on the mean holding times numerically that are derived from the steady state as described in Section 3.

5 Numerical Results

5.1 Implementation

We have implemented the method described in Section 3 and Section 4 in a *Mathematica* script [15]. It takes the input parameters as described in Section 2 and generates

Table 2. Input Parameters for Case I and CaseII

Parameter	Case I	Case II
$\Delta(1)$	$2 \cdot \Delta$	$4 \cdot \Delta$
$\Delta(2)$	$3 \cdot \Delta$	$4 \cdot \Delta$
$\Delta(3)$	$3 \cdot \Delta$	Δ
$\hat{a}(1)$	1	1
$\hat{a}(2)$	3	3
$\hat{a}(3)$	$1 \ldots 6$	$1 \ldots 6$
$\lambda(1)$	$2 \cdot \lambda$	λ
$\lambda(2)$	$1.333 \cdot \lambda$	λ
$\lambda(3)$	$1.333 \cdot \lambda$	$4 \cdot \lambda$

Table 3. Common Parameters for Case I and Case II

Parameter	Value
Δ	0.049
λ	29.16
$\mu(1) = \mu(2) = \mu(3)$	32.03

the original and the modified state space and the corresponding generator matrixes. On a 750 MHz *Compaq Armada* PC with 256 MB RAM, this script (using Mathematica's *LinearSolve* function) is able to solve systems of around 4000 states in about 15 minutes.

5.2 Input Parameters

The input parameters for the two example cases that we study are summarized in Tables 3 and 2. We let $i = 0$. Class-1 is a "rigid" class in the sense that class-1 sessions cannot be slowed down once they admitted into the system. The maximum slow down value for class-2 sessions is kept fixed at $\hat{a}(2) = 3$, while we vary $\hat{a}(3) = 1 \ldots 6$. In Case I, the class-wise load is: $\rho(k) = (\lambda(k)/\mu(k)) \cdot \Delta(k) = 4 \cdot \frac{\lambda \cdot \Delta}{\mu}$ and note that $\Delta(2) = \Delta(3)$, that is the (peak) resource requirement of the two elastic classes are the same. In Case II, the class-wise load is the same as in Case I, but now $\Delta(2) = 4 \cdot \Delta(3)$.

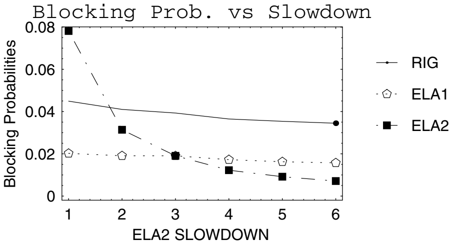


Fig. 2. Case I, Blocking Probabilities at *Fixed* Slow Down Rates: $a(2) = 3, a(3) = 1 \ldots 6$

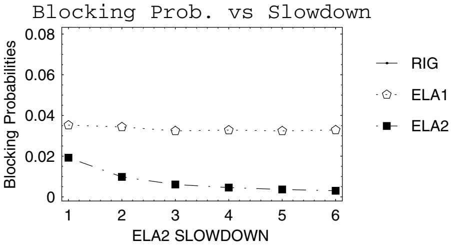


Fig. 3. Case 4, Case II, Blocking Probabilities at *Fixed* Slow Down Rates: $a(2) = 3, a(3) = 1 \ldots 6$

5.3 Blocking Probabilities

Figures 2 and 3 depict the class-wise blocking probabilities when the slow down rates are kept fixed as in [3] for Case I and II respectively. This fixed slow down rate is

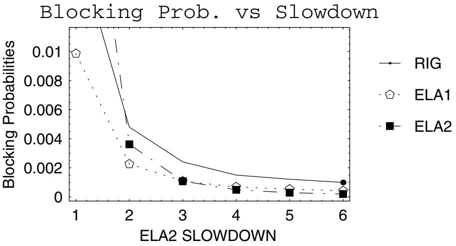


Fig. 4. Case I, Blocking Probabilities at *State Dependent* Slow Down Rates: $\hat{a}(2) = 3, \hat{a}(3) = 1 \ldots 6$

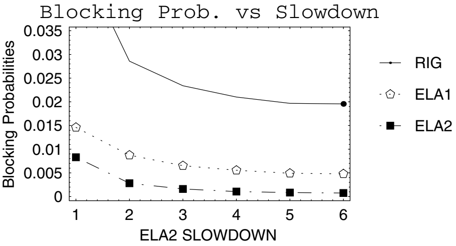


Fig. 5. Case II, Blocking Probabilities at *State Dependent* Slow Down Rates: $\hat{a}(2) = 3, \hat{a}(3) = 1 \ldots 6$

3 for class-2 (”ELA1”) and is set to $1 \ldots 6$ for class-3 (”ELA2”). In Case I, where resource requirement of the two elastic classes are equal ($\Delta(2) = \Delta(3)$), the blocking probabilities are of course also equal when the class-3 slow down rate is also set to 3. More interestingly, we see that in Case I, allowing for a higher slow down value helps to reduce the class-3 blocking probability from 8% to around 1%. The class-1 blocking probabilities are not much affected and stay around above 4% and around 12 % in the two cases.

It is noteworthy, that just as expected, the blocking rates for the system with dynamic slow down are lower than with fixed slow down. (Compare Figure 4 with Figure 2 and Figure 5 and with Figure 3.) This is of course because in the dynamic slow down case the sessions are not slowed down in the lightly loaded system states. Note also that in the fixed slow down system there is no need for a resource sharing policy, since the slow down rates are kept fixed in all system states.

5.4 Mean Time in System

Figure 6 and Figure 7 show the impact of slow down on the mean time spent by elastic flows in the system in Case I and Case II respectively. In Case I, as expected, the mean

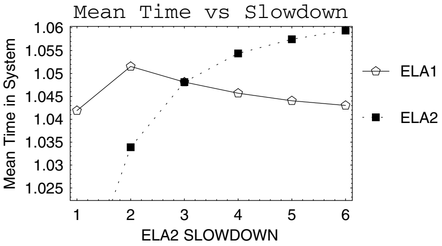


Fig. 6. Case I, Mean Time Spent in the System at *State Dependent* Slow Down Rates: $\hat{a}(2) = 3, \hat{a}(3) = 1 \ldots 6$ Normalized to $1/\mu$

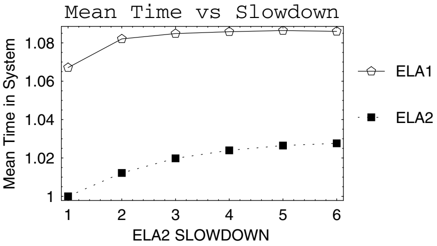


Fig. 7. Case II, Mean Time Spent in the System at *State Dependent* Slow Down Rates: $\hat{a}(2) = 3, \hat{a}(3) = 1 \ldots 6$ Normalized to $1/\mu$

times are equal when $\hat{a}(3) = \hat{a}(2) = 3$. In Case II, class-2 sessions need to transmit significantly more data than class-3 sessions, since $R_p(2) > R_p(3)$. This clarifies that class-2 sessions spend somewhat more time in the system for all $\hat{a}(3)$ values.

In Figure 6 we notice that when $\hat{a}(3)$ is set to 2, the mean time for class-2 calls increases somewhat (from roughly 4% to 5%) as compared to the original holding time $1/\mu(2)$, while the blocking probabilities decrease significantly. When $\hat{a}(3)$ is further increased, the class-2 average time spent in the system decreases ! This seemingly strange behavior is due to the fact that at $\hat{a}(3) = 2$, $\hat{a}(2)$ is still greater, and when class-3 sessions make better utilization of the resources, it is at the cost of class-2 resources. As class-3 sessions become "more and more elastic", it is the class-3 sessions that start spending more time in the system.

5.5 Second Moment of the Time Spent in the System

The second moments show similar behavior as the mean time spent in the system (Figures 8 and 9.) We also see that increasing the maximum slow down factor beyond relatively small values (i.e. beyond $\hat{a}(3) = 4$) have only a small impact as compared to the impact when setting $\hat{a}(3) = 3$.

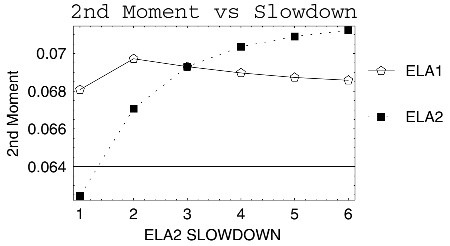


Fig. 8. Case I, Second Moment of the Time Spent in the System at *State Dependent* Slow Down Rates: $\hat{a}(2) = 3, \hat{a}(3) = 1 \ldots 6$ Normalized to $1/\mu$

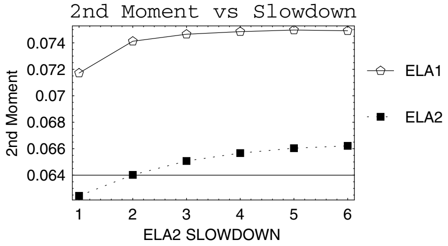


Fig. 9. Case II, Second Moment of the Time Spent in the System at *State Dependent* Slow Down Rates: $\hat{a}(2) = 3, \hat{a}(3) = 1 \ldots 6$ Normalized to $1/\mu$

6 Conclusions

In this paper we considered an uplink capacity limited CDMA cell, where multiple service classes receive service. Service classes in this model are characterized by their Poisson arrival rates and the mean value of their exponentially distributed nominal holding times. So called rigid service classes occupy a fix amount of resource when they are admitted into the system. Elastic class sessions rate and thereby instantaneously occupied resources are adaptively adjusted between some minimum and maximum value. This rather general flow level model can be seen as an extension of Altman's rate controlled CDMA model that is described in [3].

This model is analyzed in the paper by means of developing a Markov model and solving for the steady state. As it could be expected, the dynamic adjustment of the slow down rates result in significantly smaller blocking probabilities than that of the system of [3]. That is, the dynamic rate controlling system is expected to have higher Erlang capacity under the same blocking probability constraints.

References

1. A. J. Viterbi, “CDMA - Principles of Spread Spectrum Communication”, *Addison-Wesley*, ISBN 0-201-63374-4, 1995.
2. S. Glisic, B. Vucetic, “Spread Spectrum CDMA Systems for Wireless Communications”, *Artech House Publishing*, ISBN 0-89006-858-5, 1997.
3. E. Altman, “Capacity of Multi-service Cellular Networks with Transmission-Rate Control: A Queueing Analysis”, *ACM Mobicom ’02*, Atlanta, GA, September 23-28, 2002.
4. E. Altman, “Rate Control and QoS-related Capacity in Wireless Communications”, - Keynote Speech at *Quality of Future Internet Services - QoFIS*, Stockholm, October 2003.
5. A. T. Andersen, S. Blaabjerg, G. Fodor, M. Telek, “A Partially-Blocking Queueing System with CBR and ABR Arrival Streams” *Telecommunication Systems 19:1*, Kluwer Academic Publishers, pp. 75-99, 2002.
6. S. Racz, B. P. Gero, G. Fodor, “Flow Level Performance Analysis of a Multi-service System Supporting Elastic and Adaptive Services”, *Performance Evaluation 49*, Elsevier, pp. 451-469, 2002.
7. H. Holma, A. Toskala, “WCDMA for UMTS - Radio Access for Third Generation Mobile Communications”, Wiley, ISBN 0 471 72051 8, First Edition, 2000.
8. K. W. Ross, “Multiservice Loss Models for Broadband Telecommunication Networks”, ISBN 3-540-19918-8, *Springer Verlag*, 1995.
9. Eitan Altman, Damien Artiges and Karim Traore, “On the Integration of Best-Effort and Guaranteed Performance Services”, *INRIA Research Report No. 3222*, July, 1997.
10. R. Nunez Queija, J. L. van den Berg, M. R. H. Mandjes, “Performance Evaluation of Strategies for Integration of Elastic and Stream Traffic”, *International Teletraffic Congress*, UK, 1999.
11. L. Massoulie and J. Roberts, “Bandwidth Sharing and Admission Control for Elastic Traffic”, Available at: `http://www-sop.inria.fr/mistral/pub/massoulie.html`.
12. R. A. Howard, “Dynamic Probabilistic Systems and Markov Decision Processes”, *MIT Press*, 1960.
13. J. A. Buzacott, “Markov Approach to Finding Failure Times of Repairable Systems”, *IEEE Transactions on Reliability*, Vol. R19, pp. 152-156, November 1970.
14. B. F. Nielsen, “Modeling Multiple Access Systems with Phase Type Distributions”, Ph.D. Thesis, *IMSOR, Technical University of Denmark*, Thesis No. 49, 1988.
15. S. Wolfram, “The Mathematica Book”, V4.0, ISBN 0521643147, 1999.

Queue Analysis for Wireless Packet Data Traffic

Shahram Teymori and Weihua Zhuang

Centre for Wireless Communications (CWC),
Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
{steymori, wzhuang}@bbr.uwaterloo.ca

Abstract. Recent research based on Internet traffic measurements shows that the traffic flows have a fractal nature (i.e., self-similarity property), which causes the underestimation of the network engineering parameters when using the Poisson model. Preliminary field measurements demonstrate that packet data traffic in wireless communications also exhibits self-similarity. In this research, we investigate the queuing behavior of a self-similar traffic flow for wireless packet data applications. The traffic is modelled by an on-off source with heavy-tailed on periods. We derive a new relationship among the asymptotic distribution of the queue length, traffic specifications, wireless channel characteristics, and transmission rate. Computer simulation results are given to demonstrate the accuracy of the theoretical analysis.

Keywords: Loss probability, queueing analysis, self-similarity, wireless packet data.

1 Introduction

Wireless communication systems have been revolutionized by technological advances in the last decades. The third generation (and beyond) wireless networks will use the packet-switching technology to provide high-speed data services such as File Transportation Protocol (FTP) and web browsing. These applications have stringent performance requirements in terms of throughput, and transmission accuracy (packet loss rate). Data traffic flows are not sensitive to delay and can tolerate delays such as seconds and minutes, depending on the application; however, it is sensitive to packet loss probability and can normally tolerate the loss probability only up to 10^{-6}. Selection of a proper model to analyze the queuing behaviors of network traffic flows plays an important role in performing network engineering. Recently it has been shown that the conventional Poisson traffic model is not proper for packet data traffic in the Internet [1]. It is also observed that the aggregated Internet traffic flows have similar patterns in different time scales, referred to as self-similarity [2]. Similarly, the existence of self-similarity in wireless network traffic flows has been observed [3] and has attracted attention in the research community [4]. The self-similarity

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 217–227, 2005.

characteristics result in that traffic engineering techniques, based on the traditional method, underestimate the required resources (such as transmission rate) to achieve quality-of-service (QoS) satisfaction.

Willinger *et al.* proved that the superposition of a large number of on-off traffic sources, with heavy-tailed on or off periods, results in self-similar aggregated traffic [5]. The finding is important, as it indicates that heavy-tailed on-off periods can be the reason for self-similarity in TCP/IP[1] traffic. The result is confirmed in [6], which showed that individual data traffic source can be modelled by on-off sources with heavy-tailed on and off periods.

Unfortunately the network analysis with heavy-tailed traffic sources is very complicated. For example, in an M/G/1 queuing system, the average packet delay is proportional to the variance of the service time [7]. In the presence of heavy-tailed on-off sources, the service time variance is infinite, which implies an infinite value for the average delay. Furthermore, it can be shown that, for on-off sources with heavy-tail on periods, the moment generation function (MGF) is infinite [7]. This means that the Chernoff bound can not be used to analyze the queue delay performance. Jenkovic in [8] derived a relation among queue length distribution, transmission rate, and traffic characteristics for a single server system with a single on-off source in a wireline network. The analysis is carried out under the assumption that the network transmission rate (server capacity) is a constant. The assumption is not valid in wireless communications. Due to the time-variant fading dispersive propagation medium, the link capacity in a wireless system changes with time, and the transmission rate is not constant.

In this research, we study the queuing behavior of self-similar packet data traffic in a wireless single-server network, and derive a new relationship among the asymptotic distribution of queue length, traffic specifications, channel characteristics, and transmission rate. This study provides insights of the impact of the self-similar property on network resource allocation for QoS provisioning. The remainder of this paper is organized as follows. Section 2 presents mathematical definitions and describes the system model. Details of the queue analysis with self-similar input traffic is given in Section 3. Computer simulation results are presented in Section 4 to demonstrate the accuracy of the theoretical analysis. Conclusions are given in Section 5.

2 System Model

We first present some mathematical definitions and concepts, which are used in this work.

Definition 1. [8] A cumulative distribution function (CDF) F on $[0, \infty]$ is called heavy-tailed ($F \in Ł$) if

$$\lim_{x \to \infty} \frac{1 - F(x-y)}{1 - F(x)} = 1, \quad y \in \Re. \tag{1}$$

[1] Transmission Control Protocol/Internet Protocol.

Definition 2. [8] A CDF function F on $[0, \infty]$ is called subexponential ($F \in S$) if

$$\lim_{x \to \infty} \frac{1 - F^{*2}(x)}{1 - F(x)} = 2, \quad y \in \Re \tag{2}$$

where $F^{*2}(x)$ denotes the 2nd convolution of F with itself, i.e, $F^{*2}(x)$=$\int_0^{+\infty} F(x-y)F(y)dy$.

A well known example of subexponentially distributed functions is functions of regular variation R_α (in particular Pareto family); $F \in R_\alpha$ if it is given by

$$F(x) = 1 - \frac{l(x)}{x^\alpha}, \qquad \alpha \geq 0 \tag{3}$$

where $l(x) : \Re_+ \longrightarrow \Re_+$ is a function of slow variation, i.e., $\lim_{x\to\infty} \frac{l(\delta x)}{l(x)} = 1$.

In this work data traffic flows are modelled by on-off sources. An on-off source can be considered as a renewal process, with renewal periods of ($\tau_n^{\text{on}} + \tau_n^{\text{off}}$, $n \geq 0$), where τ_n^{on} and τ_n^{off} are the durations of on and off periods, respectively. The on periods follow a Pareto distribution and the off periods follow an exponential distribution. During each on period, packets are generated at a constant rate of r. The steady state probabilities of being in the on and off states are given by $P_{on} = \frac{\bar{\tau}^{\text{on}}}{\bar{\tau}^{\text{on}} + \bar{\tau}^{\text{off}}}$ and $P_{off} = \frac{\bar{\tau}^{\text{off}}}{\bar{\tau}^{\text{on}} + \bar{\tau}^{\text{off}}}$, respectively, where $\bar{\tau}^{on}$ and $\bar{\tau}^{off}$ are the average on and off periods.

Consider transmission of data packets of equal length over a wireless channel. With a constant transmission information rate, the transmission times for all the packets are the same, referred to as packet time. An important QoS parameter in data service is the transmission accuracy. Consider the transmission over a wireless channel. The transmission accuracy can be described by the transmission bit error rate (BER). Given the modulation and coding scheme, the channel model and parameters, and the receiver structure, the required BER can be mapped one-to-one to the required received signal to interference-plus-noise ratio (SINR)[9]. For a given maximum transmit power, the required SINR can not be achieved and the data transmission stops if the channel is in a poor (deep fading) condition. When the channel condition improves over time (to a non deep fading condition), the required SINR can be ensured and the transmission over the channel can resume. As a result, we use a two-state channel model. The channel is said to be in a nonworking state in the former case and in a working state in the latter case. If the channel fades slowly with respect to the packet time, the channel state remains the same over a packet time. Given the channel fading statistics, the working-state probability P_w and nonworking-state probability P_{nw} can be calculated. The transmission system for each packet traffic flow can be described by a simple queueing model, where the input is a single on-off source and the packets are transmitted through a two-state fading channel, as illustrated in Figure 1. In the working state of the channel, the server transmits packets with a constant transmission rate of C, and in the nonworking state the server stops the transmission to save the resources. The scheduling principle based on the channel condition achieves high utilization efficiency of the limited radio resources and has been used in 3G wireless system design. As data traffic is in general not strictly delay sensitive, the QoS requirement for transmission delay can be specified by two parameters, q_{th} and ϵ: the probability that the queue length Q at the scheduler exceeds the threshold q_{th} is less than a small value ϵ, i.e., $P[Q > q_{th}] < \epsilon$. Let $c(t)$ denote the time varying channel capacity to guarantee the required BER and transmission delay. Then, $c(t) = C$ if the channel is in a working state and $c(t) = 0$ if the channel is in a nonworking state, at time t. The scheduler calculates and allocates the bandwidth of $c(t)$ by considering the channel status and traffic specifications, which are assumed to be known to the scheduler.

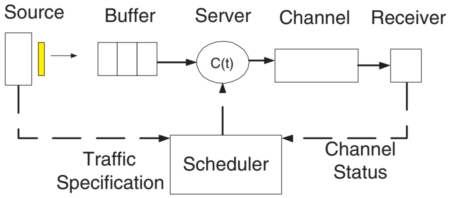


Fig. 1. Single server single input system

3 Queuing Analysis

We start the analysis by using a classical result given in [10]. Let a_n and s_n, $n \geq 0$, be two sequences of i.i.d. random variables, representing the number of packet arrival and departures in n_{th} recursion (with τ_n^{on} on period and τ_n^{off} off period of the source), respectively. Let X_n $(= a_n - s_n)$ denote the net increment of the packet number. In each recursion, the queueing process can be expressed as (Lindley recursion)

$$Q_{n+1} = (Q_n + X_n)^+, \qquad n \geq 0 \tag{4}$$

where $q^+ = \max(q, 0)$, and Q_n is the queue size at the beginning of the n^{th} recursion. According to [10], this recursion admits a unique stationary solution and, for all initial conditions, the probability $P[Q_n > x]$ converges to the stationary probability of $P[Q > x]$ under stability condition $\bar{X}_n < 0$ (where $\bar{X}_n$ denotes the expected value of X_n). In the following derivations, we assume that all the queuing systems under consideration are in their stationary regimes.

The residual life of a renewal process is defined as the duration between some fixed time t and the starting point of the following renewal. It is one of the random variables that describe the local behavior of a renewal process. Another variable is the age at time t [11], or the time already elapsed in the current renewal. When we look at a renewal process in the reverse (backward) direction, we again observe a renewal process having the same probability structure, but the residual life time is called age [11]. Letting $F(x)$ denote the CDF of X_n, the CDF of the residual life, $F1(x)$, is given by

$$F1(x) = \frac{\int_0^x (1 - F(u))du}{\bar{X}_n}. \tag{5}$$

$F1(x)$ is also called the integral tail distribution of $F(x)$.

Theorem 1 [8] If $F_X(x)$ is heavy-tailed and $F1_X(x)$ is subexponential and $\bar{X}_n < 0$, for (4) it can be shown that

$$P[Q_n > x] \sim \frac{\int_x^\infty P[X_n > u]du}{-\bar{X}_n}, \quad x \to \infty \tag{6}$$

where $l(x) \sim j(x)$ denotes $l(x)/j(x) = 1$ as $x \to \infty$.

Lemma 1 [8] Let X and Y be two independent random variables distributed as $F(x)$ and $G(x)$, respectively. If $F(x)$ is heavy tailed, $\bar{X}$ and $\bar{Y}$ are finite, and Y is a positive non-heavy-tailed random variable, then

$$P[X - Y > x] \sim P[X > x] = 1 - F(x), \quad x \to \infty. \tag{7}$$

Consider the system illustrated in Figure 1, during the interval of $\mathit{\Gamma}_n = \tau_n^{\mathrm{on}} + \tau_n^{\mathrm{off}}$, the state of the channel may change several times, so does the capacity. We define $\mathit{\Omega}_n$ (in second) as the summation of all the time intervals (during the $\mathit{\Gamma}_n$ period) when the channel is in a working state. In a working state with capacity C, if there are backlogged packets in the queue, the number of the transmitted packets is $C\mathit{\Omega}_n$. We define the effective capacity of the time varying channel as $c_n = \frac{C\mathit{\Omega}_n}{\mathit{\Gamma}_n}$. Instead of considering the time-varying capacity, the effective capacity that is constant (even though random) during $\mathit{\Gamma}_n$ will be considered. As the PDF of c_n, denoted by $f_{c_n}(x)$, is unknown, we need to estimate the value of c_n to proceed further.

Consider the evolution of the queue length at the initial moment of the on period, denoted by Q_n^p, where the superscript p stands for Palm probability (meaning that the queue length is observed at the beginning of the on period). According to the definition of X_n, it can be shown that $X_n = r\tau_n^{on} - C\mathit{\Omega}_n$, and we have

$$\begin{aligned} Q_{n+1}^p &= (Q_n^p + X_n)^+ \\ &= (Q_n^p + r\tau_n^{\mathrm{on}} - C\mathit{\Omega}_n)^+ \\ &= (Q_n^p + r\tau_n^{\mathrm{on}} - c_n(\frac{\mathit{\Gamma}_n}{\mathit{\Omega}_n})\mathit{\Omega}_n)^+ \\ &= (Q_n^p + r(\tau_n^{\mathrm{on}}) - c_n(\tau_n^{\mathrm{on}} + \tau_n^{\mathrm{off}}))^+ \\ &= (Q_n^p + \tau_n^{\mathrm{on}}(r - c_n) - c_n\tau_n^{\mathrm{off}})^+ \end{aligned}$$

where $r - c_n > 0$.

With a Pareto distributed on period and an exponentially distributed off period, we have

$$P[\tau_n^{\mathrm{on}} > x] = \frac{b}{x^\alpha}, \quad x \geq 0$$

and

$$P[\tau_n^{\mathrm{off}} > x] = e^{-\lambda x}, \quad x \geq 0.$$

By Lemma 1, for $x \to \infty$, we have

$$\begin{aligned} P[X_n > x] &= P[\tau_n^{\text{on}}(r - c_n) - c_n \tau_n^{\text{off}} > x] \\ &\sim P[\tau_n^{\text{on}}(r - c_n) > x]. \end{aligned} \tag{8}$$

Since

$$P[X_n > x] = E_{c_n}(P[X_n > x | c_n]) \tag{9}$$

we have

$$\begin{aligned} P[X_n > x] &\sim E_{c_n}(P[\tau_n^{\text{on}}(r - c_n) > x | c_n]) \\ &= E_{c_n}[\frac{b(r - c_n)^{\alpha}}{x^{\alpha}}] \\ &= \frac{bk}{x^{\alpha}} \end{aligned} \tag{10}$$

where $k = E_{c_n}[(r - c_n)^{\alpha}]$. Equation (10) shows that X_n also has a Pareto distribution. Under the assumption that the on and off periods are stationary, it can be shown that

$$\begin{aligned} \bar{X}_n &= E_{c_n}(\bar{\tau}_n^{\text{on}}(r - c_n) - c_n \bar{\tau}_n^{\text{off}}) \\ &= r\bar{\tau}^{\text{on}} - (\bar{\tau}^{\text{on}} + \bar{\tau}^{\text{off}})\bar{c}_n. \end{aligned} \tag{11}$$

Assuming that c_n is an ergodic process, we have $\bar{c}_n = CP_w$ and

$$\bar{X}_n = r\bar{\tau}^{\text{on}} - (\bar{\tau}^{\text{on}} + \bar{\tau}^{\text{off}})CP_w. \tag{12}$$

Considering the stability condition $\bar{X}_n < 0$, it is required that

$$\frac{r\bar{\tau}^{\text{on}}}{\bar{\tau}^{\text{on}} + \bar{\tau}^{\text{off}}} < CP_w. \tag{13}$$

As a large queue length is mainly the result of large Γ_n values (large bursts), the probability of $P[Q > x]$ for a large x depends mainly on large Γ_n values. When Γ_n is large, we can assume that $\frac{\Omega_n}{\Gamma_n} \simeq P_w$ and the effective capacity $c_n \simeq CP_w$. In the case of a large queue, X_n can be approximately estimated by

$$X_n \simeq \tau_n^{\text{on}}(r - CP_w) - CP_w \tau_n^{\text{off}} \tag{14}$$

for large X_n values. Due to the fact that the distribution of X_n is heavy tailed, by Theorem 1, it can be shown that

$$P[Q^p > x] \sim \frac{bk}{[CP_w(\bar{\tau}^{\text{on}} + \bar{\tau}^{off}) - r\bar{\tau}^{\text{on}}](\alpha - 1)x^{\alpha - 1}}, \quad x \to \infty. \tag{15}$$

Substituting $k = E_{c_n}(r - c_n)^{\alpha} \simeq (r - CP_w)^{\alpha}$,

$$P[Q^p > x] \approx \frac{b(r - CP_w)^{\alpha}}{[CP_w(\bar{\tau}^{on} + \bar{\tau}^{\text{off}}) - r\bar{\tau}^{on}](\alpha - 1)x^{\alpha - 1}}, \quad x \to \infty \tag{16}$$

where $l(x) \approx j(x)$ denotes $l(x)/j(x) \simeq 1$ as $x \to \infty$. The stationary probability $P[Q > x]$ can be calculated from the Palm queue in (16).

Let $\Lambda_n^{\text{on}}, \ -\infty < n < \infty$, be a random process that represents the beginning of the on periods in the stationary on-off process with $\Lambda_0^{on} < 0 \leq \Lambda_1^{\text{on}}$, and B be a Bernoulli random variable with $P(B=0) = 1 - P(B=1) = \frac{\bar{\tau}^{on}}{\bar{\tau}^{\text{on}} + \bar{\tau}_{off}}$. We define the residual on period, $\tau_n^{on,r}$, as the residual life time of τ_n^{on} with respect to time 0. Similarly, the residual off period, $\tau_n^{off,r}$, can be defined. Λ_0^{on} can be represented by

$$-\Lambda_0^{\text{on}} = B(\tau_0^{on} + \tau_0^{\text{off,r}}) + (1-B)\tau_0^{\text{on,r}}. \qquad (17)$$

The preceding equation means that, if the source is in the on state ($B = 0$) at time $t = 0$, $-\Lambda_0^{\text{on}}$ is the age (residual life) of the on period, $\tau_0^{\text{on,r}}$; otherwise, $-\Lambda_0^{\text{on}}$ is the duration of the on period plus the age (residual life) of the off period. Furthermore, the net increment X_0 of the load that arrives to the queue in the interval $[\Lambda_0^{\text{on}}, 0]$ is equal to

$$X_0 = B[\tau_0^{on}(r - c_0) - c_0\tau_0^{\text{off,r}}] + (1-B)[(r-c_0)\tau_0^{\text{on,r}}]. \qquad (18)$$

Let Q_{T_0} denote the backlogged traffic observed at the beginning of the zeroth renewal period ($t = \Lambda_0^{on}$). The asymptotic probability of the queue length in the zeroth renewal period, $P[Q_0 > x]$, is

$$\begin{aligned} &P[Q_0 > x] \\ &= P[Q_0 > x, B = 1]P[B = 1] + P[Q_0 > x, B = 0]P[B = 0] \\ &= P[Q_{T_0} + (r - c_0)\tau_0^{\text{on}} - c_0\tau_0^{\text{off,r}} > x]P[B = 1] \\ &\qquad + P[Q_{T_0} + (r - c_0)\tau_0^{\text{on,r}} > x]P[B = 0]. \qquad (19) \end{aligned}$$

Note that Q_{T_0} is equal to the Palm queue Q_0^p. Since Q_{T_0} and $\tau_0^{\text{on,r}}$ are subexponentially distributed, it can be shown that [8]

$$[Q_{T_0} + (r - c_0)\tau_0^{\text{on,r}} > x] \sim P[Q_{T_0} > x] + P[(r - c_0)\tau_0^{\text{on,r}} > x], \ x \to \infty.(20)$$

Also by Lemma 1 [8],

$$P[Q_{T_0} + (r - c_0)\tau_0^{\text{on}} - c_0\tau_0^{\text{off,r}} > x] \sim P[Q_{T_0} > x], \quad x \to \infty. \qquad (21)$$

Equation (19) can then be simplified by using (20) and (21),

$$P[Q_0 > x] \sim P[Q_{T_0} > x] + P[(r - c_0)\tau_0^{\text{on,r}} > x]P(B = 0), \ x \to \infty. \quad (22)$$

Considering that $P[\tau_0^{\text{on}} > x] = \frac{b}{x^\alpha}$ for $x \geq 0$, we have, for $x \geq 0$,

$$P[(r - c_0)\tau_0^{\text{on,r}} > x] = E_{c_0}\left(\frac{\int_{\frac{x}{r-c_0}}^{\infty} \frac{b}{u^\alpha} du}{\bar{\tau}_0^{\text{on}}}\right) = E_{c_0}\left(\frac{b(r - c_0)^{\alpha - 1}}{\bar{\tau}_0^{on}(\alpha - 1)x^{\alpha - 1}}\right). \qquad (23)$$

Using $c_0 \simeq CP_w$, we have

$$P[(r-c_0)\tau_0^{\text{on,r}} > x] \simeq \frac{b(r-CP_w)^{\alpha-1}}{\bar{\tau}_0^{on}(\alpha-1)x^{\alpha-1}}. \tag{24}$$

With $Q_{T_0} = Q_0^p$ and Q being stationary (i.e., $P[Q_0 > x] = P[Q_n > x] = P[Q > x]$), applying (15) and (24) to (22), we have

$$\begin{aligned} P[Q > x] &\approx P[Q_0^p > x] + (\frac{1}{\bar{\tau}^{\text{on}} + \bar{\tau}^{\text{off}}})\frac{b(r-CP_w)^{\alpha-1}}{(\alpha-1)x^{\alpha-1}} \\ &\approx \frac{b(r-CP_w)^{\alpha}}{(CP_w(\bar{\tau}^{on}+\bar{\tau}^{\text{off}}) - r\bar{\tau}^{on})(\alpha-1)x^{\alpha-1}} \\ &\quad + \frac{b(r-CP_w)^{\alpha-1}}{(\bar{\tau}^{on}+\bar{\tau}^{\text{off}})(\alpha-1)x^{\alpha-1}}, \quad x \to \infty. \end{aligned} \tag{25}$$

In (25), we estimate the asymptotic behavior of Q for a large x. By using this relationship, we are able to calculate the required link capacity (transmission rate) C based on traffic parameters (r, $\bar{\tau}_n^{\text{on}}$, $\bar{\tau}_n^{\text{off}}$, b, α) and channel characteristics (P_w) to reach a predefined QoS level (in terms of delay specified by q_{th} and ϵ).

4 Simulation

Due to the complexity of the queue analysis with the self-similar input traffic flow, we have to make several simplified assumptions for tractability in deriving (25). To validate the analysis given in Section 3, computer simulations were carried out and the results are presented in the following.

The single server single input system illustrated in Figure 1 is simulated. The information bits of the input traffic flow are generated during each on period at a rate of r. The on periods are determined from the sample values of the Pareto random variable, generated by using the random generator as specified in [12]. The off periods are determined from the sample values of the exponential random variable. As the data packets have the same length and are transmitted at a constant rate, without loss of generality, we simulate the generation and transmission of bit flows instead of packet flows. The simulation parameters are chosen based on the previous measurements for FTP application in wireless networks [13] to be $\bar{\tau}^{\text{on}} = 0.256$ s, $\bar{\tau}^{\text{off}} = 15$ s, $r = 64$ kbps, and frame duration of 10 ms. Although $\alpha = 1.1$ is suggested in [13], we choose $\alpha = 1.4$ in the simulation, because the accuracy of the Pareto random generator is relatively poor for a small α value. The two-state channel status is characterized by a Bernoulli random process with working probability (P_w), changes independently from frame to frame. Each run in the simulations consists of at least 10e8 frames.

Figure 2 shows the tail distribution of the queue length for the buffer, with P_w being 0.5 and 0.9 respectively and capacity C of 40 kbps. The analytical results are obtained from (25). It is observed that the simulation results agree very well with the analytical results.

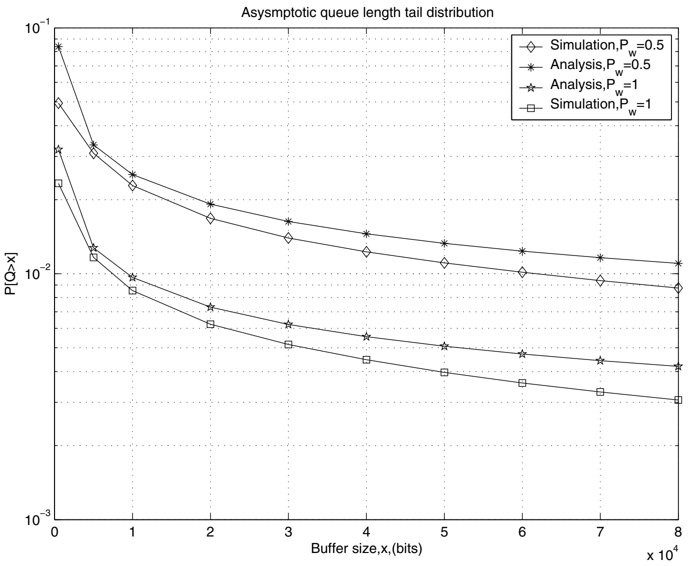


Fig. 2. Effect of the wireless channel quality on the queue length distribution

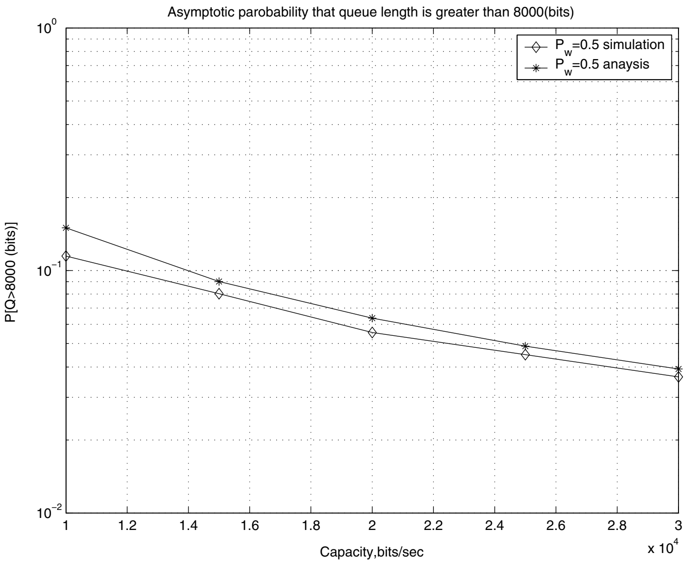


Fig. 3. Effect of the capacity on the queue length distribution

Figure 3 plots the probability, $P[Q > 8000 \text{ (bits)}]$, as a function of the system capacity C with P_w being 0.5. Again, we observe a close agreement between the analytical results obtained from (25) and the simulation results.

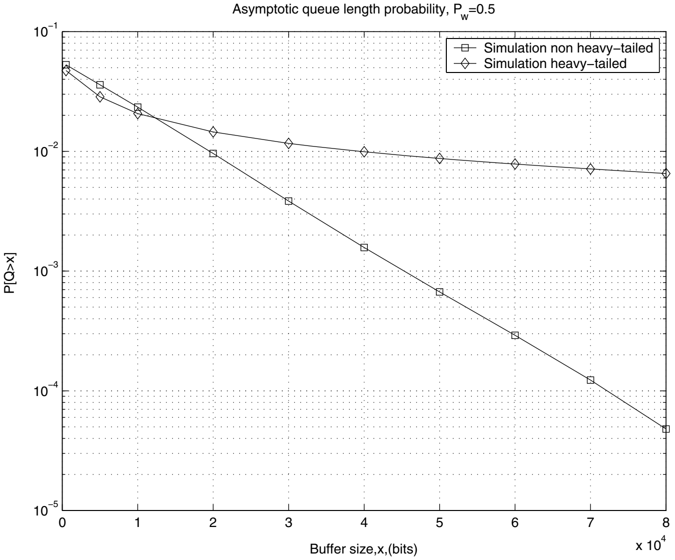


Fig. 4. Impact of the traffic model on the queue length distribution

Figure 4 shows the impact of the heavy-tailed traffic on the asymptotic tail-distribution of queue length. Two traffic flows, with the same average rate and equal expected on and off periods, are simulated respectively. One is a non heavy-tailed on-off source, where both on and off periods are exponentially distributed. The other is a heavy-tailed traffic, whose on periods have a Pareto distribution with $\alpha = 1.4$. The system parameters are the same for both inputs with $C = 40$ kbps and $P_w = 0.5$. The distribution for the non heavy-tailed traffic declines very fast (exponentially), but not in the case of the heavy-tailed source. It is the main and the most important impact of the heavy-tailed traffic flows on network performance. It is observed that the distribution for the heavy-tailed traffic has a relatively large value even for a very large queue length, which causes the possibility of a very large delay and a large probability of packet loss.

5 Conclusions

This paper investigates the impact of self-similarity property of the source traffic flow on queue length distribution in a wireless system having a single input and a single server. The wireless channel is characterized by an i.i.d two-state model. The close-form expression is derived for the relation among the traffic parameters, the channel working state probability, the server capacity, and the queue length distribution. The derivation extends the research presented in [8] for wireline networks to a wireless communication environment. The preliminary simulation results demonstrate that the assumptions made in the analysis are reasonable and the derived relation given in (25) is accurate.

Acknowledgement

This research was supported by Bell University Labs (BUL) and Communications Information Technology Ontario (CITO).

References

1. V. Paxson and S. Floyd, "Wide area traffic: the failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, June 1995.
2. M. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 835–846, December 1997.
3. M. Jiang, M. Nikolic, S. Hardy, and L. Trajkovic, "Impact of self-similarity on wireless data network performance," in *Proc. IEEE ICC'01*, vol. 2, pp. 477 –481, June 2001.
4. M. Cheng and L.F. Chang, "Wireless dynamic channel assignment performance under packet data traffic", *IEEE J. Select. Areas Commun.*, vol. 17, no. 7, pp. 1257-1269, July 1999.
5. W. Willinger, M. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at source level at source level," *IEEE/ACM Trans. Networking*, vol. 5, pp. 71–86, February 1997.
6. K. Park, G. Kimy, and M. Crovellaz, "On the relationship between file sizes, transport protocols, and self-similar network traffic," *Proc. of the Fourth International Conference on Network Protocols (ICNP'96)*, October 1996.
7. P. Jelenkovic, "Heavy tails and fluid queues: lecture notes." http://comet.ctr.columbia.edu/ anar/e6762/e6762.htm, 2002.
8. P.R. Jelenkovic and A.A. Lazar, "Multiplexing on-off sources with sub exponential on periods: part1," in *Proc. IEEE/ACM Infocom'97*, 1997.
9. B. Collins and R. Cruz, "Transmission policies for time varying channels with average delay constraints," in *Proc. 1999 Alleron Conf. Communication, Control, & Computer*, 1999.
10. R.M. Loynes, "The stability of a queue with non independent inter arrival and service time," in *in Proc. Cambridge Philosophy Society*, vol. 58, pp. 496–520, 1968.
11. S.M. Ross, *Introduction to Probability Models*, 8th ed. Academic Press, 2003.
12. G. Kramer, "Self-similar network traffic," Dept. of Computer Science University of California, http://wwwcsif.cs.ucdavis.edu/~kramer/research.html, 2001.
13. G. Iacovoni, "Wg2 interaction," Tech. Rep., Ericsson Lab, Italy, 1998.

Modeling a Beacon Enabled 802.15.4 Cluster with Bidirectional Traffic

Jelena Mišić*, Shairmina Shafi, and Vojislav B. Mišić

Department of Computer Science, University of Manitoba,
Winnipeg, Canada

Abstract. We analyze the performance of an IEEE 802.15.4 compliant network cluster operating in the beacon enabled mode with both downlink and uplink traffic. We investigate the non-saturation regime and outline the conditions under which the network abruptly goes to saturation. The operation of the WPAN is modeled through discrete time Markov chains and the theory of M/G/1 queues. The model considers acknowledged transmissions and includes the impact of different network and traffic parameters such as the packet arrival rate, packet size, inactive period between the beacons, and the number of stations. We analyze the stability of the network queues and show that the stability of the downlink queue at the coordinator is the most critical for network operation.

1 Introduction

The success of wireless sensor networks as a technology rests on the success of the standardization efforts to unify the market and avoiding the proliferation of proprietary, incompatible protocols that, although, perhaps optimal in their individual market niches, will limit the size of overall wireless sensor market [1]. The recent IEEE 802.15.4 standard for low rate wireless personal area networks is considered as one of the technology candidates for wireless sensor networks [2, 1] since it supports small, cheap, energy-efficient devices operating on battery power that require little infrastructure to operate, or none at all.

In an IEEE 802.15.4-compliant WPAN, a central controller device (commonly referred to as the PAN coordinator) builds a WPAN with other devices within a small physical space known as the personal operating space. Two topologies are supported: in the star topology network, all communications, even those between the devices themselves, must go through the PAN coordinator. In the peer-to-peer topology, the devices can communicate with one another directly – as long as they are within the physical range – but the PAN coordinator must be present nevertheless. The standard also defines two channel access mechanisms, depending on whether a beacon frame (which is sent periodically by the PAN coordinator) is used to synchronize communications or not. Beacon

* This research is partly supported by the NSERC Discovery Grant.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 228–239, 2005.

enabled networks use slotted carrier sense multiple access mechanism with collision avoidance (CSMA-CA), while the non-beacon enabled networks use simpler, unslotted CSMA-CA.

In this work, we model the 802.15.4 sensor network (cluster) with downlink and uplink transmissions, which operates in beacon-enabled mode with slotted CSMA-CA communication. This setting corresponds well to sensing applications with hierarchical topology wherein individual nodes communicate with the PAN (cluster) coordinators only. The goal of this work is to evaluate the performance of such networks, identify possible performance bottlenecks and quantify their impact. We combine the theory of discrete-time Markov chains and the theory of M/G/1 queues to derive the probability distributions of packet service times, which is then validated through simulation. We also derive the stability limits of individual queues.

To the best of our knowledge, this is the first analytical model of 802.15.4 networks with bidirectional traffic; the only other paper that considers the similar problem [4] is based exclusively on simulation results. The current work significantly extends our previous work on uplink channel modeling [5] in which downlink data transmissions were not considered at all.

The paper is organized as follows. Section 2 explains some basic features of the 802.15.4 MAC, including the CSMA-CA algorithm. Section 3 describes the analytical model of the MAC layer, which is then used to model the behavior of a complete node in Sect. 3.1. Section 4 presents and discusses the results of our analysis. Section 5 concludes the paper.

2 Basic Properties of the 802.15.4 MAC

In beacon enabled networks, the channel time is divided into superframes which are bounded by beacon transmissions from the coordinator, as shown in Fig. 1 [3]. All communications in the cluster take place during the active portion of the superframe, the duration of which is referred to as the superframe duration *SD*. During the (optional) inactive portion, nodes may enter a low power mode, or engage in other activities at will.

The active portion of each superframe is divided into equally sized slots; the beacon transmission commences at the beginning of slot 0, and the contention access period (CAP) of the active portion starts immediately after the beacon. Slots are further subdivided into backoff periods, the basic time units of the MAC protocol to which all transmissions must be synchronized. The actual duration of the backoff period depends on the frequency band in which the 802.15.4 WPAN is operating: 868 to 868.6MHz, 902 to 928MHz, or 2400 to 2483.5MHz [3]. The maximum data rates for these bands are 20kbps, 40kbps, and 250kbps, respectively.

A part of the active portion of the superframe may be reserved by the PAN coordinator for dedicated access by some devices; this part is referred to as the contention-free period (CFP), while the slots within are referred to as the

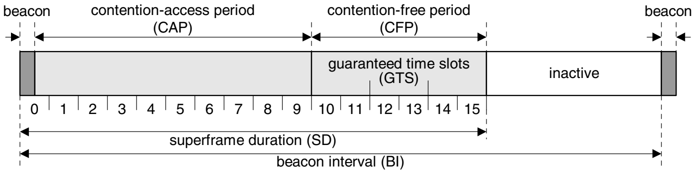


Fig. 1. The composition of the superframe under IEEE Std 802.15.4 (adapted from [3])

guaranteed time slots (GTS). In this work we do not consider the GTS, although their presence will clearly decrease the usable bandwidth of the PAN for other devices.

2.1 The CSMA-CA Algorithm

During the CAP period, individual nodes access the channel using the CSMA-CA algorithm. The algorithm begins by initializing i to zero and c to 2; the variable $i = 0 \,.\,.\, m$ (where $m = macMaxCSMABackoff - 1$) represents the index of the backoff attempt, while the variable $c = 0, 1, 2$ represents the index of the Clear Channel Assessment (CCA) phase counter. Note that the standard denotes these variables with NB and CW, respectively [3]; we use different notation in order to simplify the mathematical expressions in our model.

If the device operates on battery power, as indicated by the attribute *macBattLifeExt*, the parameter BE (the backoff exponent which is used to calculate the number of backoff periods before the node device attempts to assess the channel) is set to 2 or to the constant *macMinBE*, whichever is less; otherwise, it is set to *macMinBE* (the default value of which is 3).

The algorithm then locates the boundary of the next backoff period; as mentioned above, all operations must be synchronized to backoff time units.

In step (2), the algorithm generates a random waiting time k in the range $0 \,.\,.\, 2^{BE} - 1$ backoff periods. The value of k is then decremented at the boundary of each backoff period. Note that the counter will be frozen during the inactive portion of the beacon interval, and the countdown will resume when the next superframe begins.

When this counter becomes zero, the device must make sure the medium is clear before attempting to transmit a frame. This is done by listening to the channel to make sure no device is currently transmitting. This procedure, referred to as Clear Channel Assessment (CCA), has to be done in two successive backoff periods.

If the channel is found busy at the second CCA, the algorithm simply repeats the two CCAs starting from step (3). However, if the channel is busy at the first CCA, the values of i and BE are increased by one, while c is reset to 2, and another random wait is initiated; this is step (4) in the flowchart. In this case, when the number of retries is below or equal to *macMaxCSMABackoffs* (the default value of which is 5), the algorithm returns to step (2), otherwise it terminates with a channel access failure status. Failure will be reported to the higher protocol layers, which can then decide whether to re-attempt the transmission as a new packet or not. In our model, we assume that the transmission will be re-attempted until the final success.

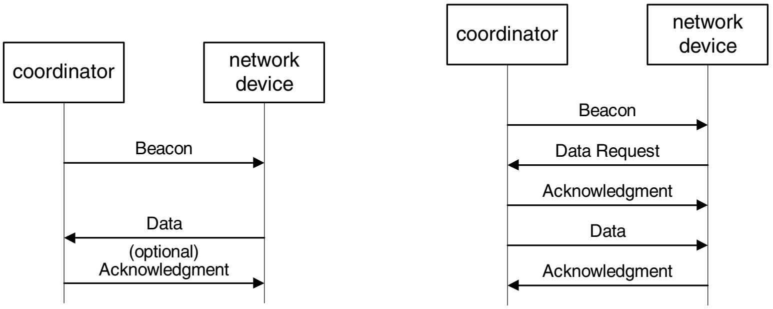
If both CCAs report that the channel is idle, packet transmission may begin.

Before undertaking step (3), the algorithm checks whether the remaining time within the CAP area of the current superframe is sufficient to accommodate the CCAs, the data frame, the proper interframe spacing, and the acknowledgment. If this is the case, the algorithm proceeds with step (3); otherwise it will simply pause until the next superframe, and resume step (3) immediately after the beacon frame.

2.2 On Uplink and Downlink Communication

According to the 802.15.4 standard, uplink data transfers from a node to the coordinator are synchronized with the beacon, in the sense that both the original transmission and the subsequent acknowledgment must occur within the active portion of the same superframe, as shown in Fig. 2(a). Uplink transmissions always use the CSMA-CA mechanism outlined above.

Data transfers in the downlink direction, from the coordinator to a node, are more complex, as they must first be announced by the coordinator. In this case, the beacon frame will contain the list of nodes that have pending downlink packets, as shown in Fig. 2(b). When the node learns there is a data packet to be received, it transmits a MAC command requesting the data. The coordinator acknowledges the successful reception of the request by transmitting an acknowledgement. After receiving the acknowledgement, the node listens for the actual data packet for the period of *aMaxFrameResponseTime*, during which the coordinator must send the data frame.



(a) Uplink transmission.

(b) Downlink transmission.

Fig. 2. Uplink and downlink data transfers in beacon enabled PAN

According to the standard, it is allowed to send the data frame 'piggy-backed' after the request acknowledgment packet, i.e., without using CSMA-CA. However, two conditions have to be fulfilled: the coordinator must be able to commence the transmission of the data packet between *aTurnaroundTime* and *aTurnaroundTime* + *aUnitBackoffPeriod*, and there must be sufficient time in the CAP for the message, appropriate inter-frame spacing, and acknowledgement; if either of these is not possible, the data frame must be sent using the CSMA-CA mechanism [3]. While the first condition depends on the implementation platform, the second depends on the actual traffic; thus *some* data frames will have to be sent using CSMA-CA. For uniformity, our model adopts a more generic approach by assuming that slotted CSMA-CA is used for all downlink transmissions, although the case where CSMA-CA is not used could be accommodated with ease. Furthermore, downlink transmissions that do not use the CSMA-CA mechanism would cause additional collisions and thus lead to the deterioration of network performance.

While the use of acknowledgment is optional (i.e., it is sent only if explicitly requested by the transmitter), in this work we assume that all the transmissions are acknowledged. In this case, the receiving node must acknowledge successful reception of the data frame within a prescribed time interval, otherwise the entire procedure (starting from the announcement through the beacon frame) has to be repeated.

According to the Section 7.5.6.4.2 of the 802.15.4 standard [3], the transmission of an acknowledgement frame shall commence at the backoff period boundary between *aTurnaroundTime* and *aTurnaroundTime* + *aUnitBackoffPeriod* after the data frame, which amounts to a delay of 12 to 32 symbol periods. Since one backoff period takes 20 symbols, this time interval may include at most one backoff period at which the channel will be assessed idle. However, a node that has finished its random countdown will need at least two CCAs before attempting transmission: while the first one may find the medium idle in between the data frame and the acknowledgment, the second one will coincide with the acknowledgment and cause the CSMA-CA algorithm to revert to the next iteration of the backoff countdown. Consequently, the acknowledgment packet cannot possibly collide with the data packet sent by another node.

It should be noted that the Section 7.5.6.7 of the standard stipulates that the data packet originator should wait for an acknowledgment for at most *macAckWaitDuration*, which amounts to 54 or 120 symbols, depending on the actual channel number. If the acknowledgment packet is not received within *macAckWaitDuration* after the original data frame, the originator may safely assume that the frame has been lost and initiate re-transmission.

2.3 Operational States

From the discussions presented above, the following states can be identified for the PAN coordinator node:

1. The coordinator may be transmitting the beacon.
2. The coordinator may be listening to its nodes and receiving data or request packets.

3. The coordinator may be transmitting the downlink data packet as a result of previously received request packet. As soon as downlink transmission is finished coordinator switches to the listening mode.

Similarly, an arbitrary (non-coordinator) node in the cluster can be in one of the following states:

1. The node may be transmitting an uplink data packet.
2. The node may be transmitting an uplink request packet.
3. The node may be in an uplink request synchronization state, which is a virtual state that lasts from the moment of new downlink packet arrival at the coordinator (or the failure of the previous downlink reception) up to the beginning of the CSMA-CA procedure for the uplink request. Note that the arrivals of downlink packets at the coordinator follow the Poisson process, whereas the corresponding announcements in the beacon (from which the target node finds out about those packets) do not.
4. The node may be waiting for a downlink packet.
5. The node may also be in an idle state, without any downlink or uplink transmission pending or in progress.

3 Basic Analytical Model

Since the same CSMA-CA algorithm is used for uplink data transmission, uplink request transmission and downlink data transmission, we will model this algorithm first, and then use it as a building block to model the operation of the node.

The MAC parameter *BO* (which stands for *macBeaconOrder*) determines the period between the beacons as $BI = 2^{BO}$ *aBaseSuperframeDuration*. For simplicity we assume that *BO* has a constant value of zero, in which case the superframe duration is $SD =$ *aBaseSuperframeDuration*; the extension of the model to accommodate different lengths of the superframe is straightforward. Note that the beacon interval and the duration of the superframe are determined by the energy management policy of the network; however, issues related to energy management are beyond the scope of the present work.

The discrete-time Markov chain for the 802.15.4 CSMA-CA algorithm is presented in Figs. 3 and 4. The 'delay line' models the case in which the remaining time within the superframe does not suffice for two CCAs, packet transmission, and reception of the acknowledgment. We assume that this Markov chain, together with the higher level structure into which it is incorporated, has stationary distribution. The process $\{i, c, k, d\}$ defines the state of the device at backoff unit boundaries. Note that the last tuple member d denotes the index of the state within the delay line mentioned above; in order to reduce notational complexity, it will be shown only within the delay line and omitted in other cases, where its value is, in fact, undefined.

Transitions between the states on Fig. 3 depend on several probabilities. First, all transitions occur at the edge of the *aUnitBackoffPeriod*. α is the probability

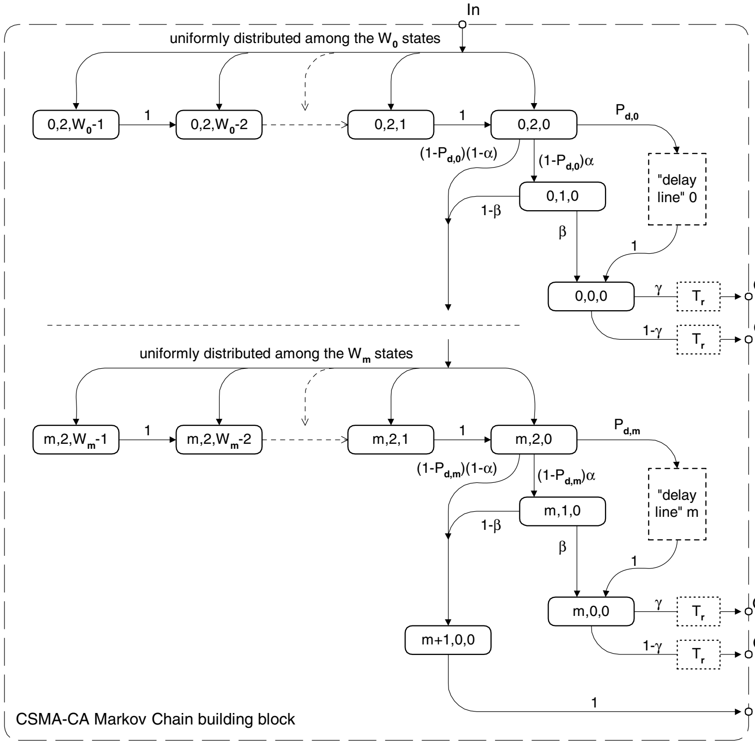


Fig. 3. General Markov chain model of the slotted CSMA-CA algorithm representing a non-idle state of the node

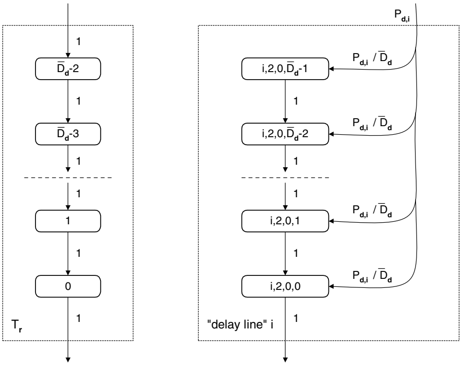


Fig. 4. Delay lines for Fig. 3

that medium is idle on the first CCA while β is the probability that medium is idle on second CCA. γ is the probability that transmission will be successful and $P_{d,i}$ is the probability that after i-th backoff attempt there will be no space

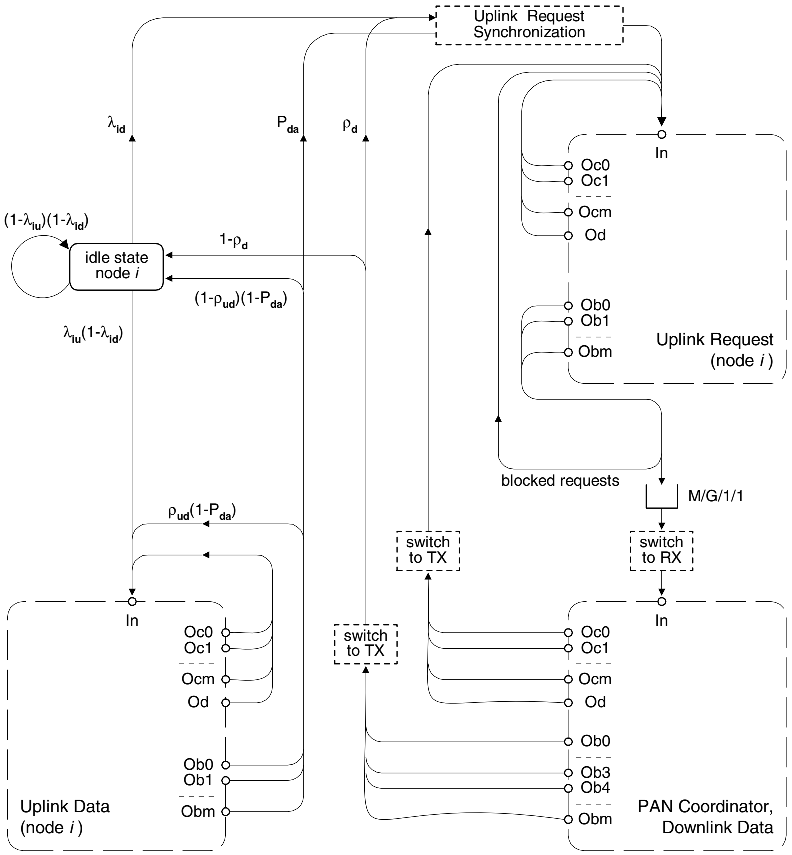


Fig. 5. Markov chain model of a node

in the current superframe to conduct two CCAs and transmission. Due to the lack of space we don't present the equations which correspond to this Markov chain.

3.1 Markov Chain Model for a Node

Let us now consider a cluster with n identical devices. The packet queues in the device data buffer and request buffer are modeled as a $M/G/1$ queueing system, in which the packet request queue has non-preemptive priority over the data queue at the device. Both uplink and downlink packet arrivals follow a Poisson process with the average arrival rate of λ_{iu} and λ_{id}, respectively.

Fig. 5 shows the high-level states of a network node – namely, the idle state, uplink data transmission, uplink request transmission and waiting for downlink data from the coordinator. As all three high level states which involve backoff procedures follow the same algorithm from Fig. 3, we have included it as a block.

After a successful uplink or downlink transmission, the node enters the idle state if both downlink and uplink data queues for the device are empty. The node will leave the idle state upon the arrival of a packet to either queue during the current backoff period. In case of simultaneous packet arrival to both queues, the downlink transmissions have priority over the uplink ones, and the node will enter the uplink request synchronization state.

Each downlink transmission must be preceded by successful transmission of a data request packet. Those packets may experience collisions, or they may arrive while the coordinator is executing backoff countdown and thus will be ignored. Thus, the behavior of the coordinator corresponds to the $M/G/1/1$ model. Upon receipt of a request, the coordinator will acknowledge it; the absence of acknowledgment means that the node must repeat the request transmission procedure.

If the downlink transmission was successful and the downlink queue towards the node is not empty, node will start a new downlink transmission cycle. If the downlink queue was empty but the uplink queue contained a packet, the node will initiate the uplink transmission cycle. Due to the priority considerations, the uplink data transmission will be started only if the downlink data queue is empty. If there was a downlink packet arrival during the uplink transmission, then as soon as the uplink transmission was finished, the node will synchronize with the beacon and attempt transmission of a request packet.

The performance descriptors for the high-level node states related to transmission are offered loads to uplink data, uplink request and downlink data queue. The offered load for the uplink data queue of the device i is denoted with $\rho_{ud} = \lambda_{iu}\overline{T_{ud}}$, where $\overline{T_{ud}}$ is the mean uplink data packet service time. The offered load for the uplink data queue of the device i is denoted with $\rho_{ur} = \lambda_{iu}\overline{T_{ur}}$ where $\overline{T_{ur}}$ is the mean request packet service time. The offered downlink load towards one node is $\lambda_{id}\overline{T_{dd}}$ where $\overline{T_{dd}}$ is the mean downlink service time.

Due to the lack of space we don't show detailed solution of the overall Markov chain. However, we argue that equations for three blocks which corrspond to uplink data, downlink data and uplink request have similar form and that the sum of state probabilities for each transmission block can be derived as a function of the idle probability x^z. If we denote the sums of state probabilities for transmission blocks as Σ^{ud}, Σ^{ur}, andΣ^{dd} respectively and the sume of state probabilities in the beacon synchronization block as Σ^s then the normalization condition for one node becomes: $x^z + \Sigma^{ud} + \Sigma^{ur} + \Sigma^{dd} + \Sigma^s = 1$ The probability to access the medium by an uplink request from a node is $\tau_{ur} = \sum_{i=0}^{m} x_{i,0,0}^{ur}$ By the same token, the probability to access the medium by an uplink data packet from node i is $\tau_{ud} = \sum_{i=0}^{m} x_{i,0,0}^{ud}$ while the probability to access the medium from the coordinator towards node i is $\tau_{dd} = \sum_{i=0}^{m} x_{i,0,0}^{dd}$ The total probability of uplink access for one node is $\tau_u = \tau_{ur} + \tau_{ud}$. The total probability of downlink access by the coordinator is $\tau_{dtot} = \tau_{dd}(1 + (n-1)\rho_{ur})$, where the first term corresponds to the target node while the second corresponds to the background traffic with the remaining $n-1$ nodes.

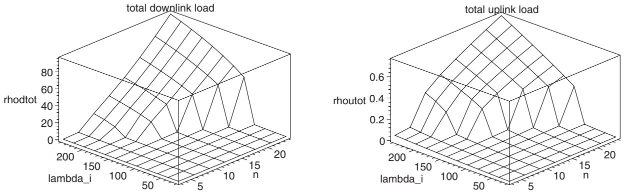
Stability of a queueing system means that the mean number of packets serviced is not smaller than the mean number of packets entered; if this is not the case, packet delays will experience inordinate growth and the system will effectively cease to operate. In an 802.15.4 cluster, the stability requirement translates into the following conditions. First, the total offered load entering the downlink queue at the coordinator (which is $\rho_{dtot} = n\rho_{dd}$) cannot exceed 1. Second, the sum of the offered uplink loads per node $\rho_{ud} + \rho_{ur}$ has to be smaller than 1.

Adherence to these conditions is reflected through the access delays for uplink and downlink traffic. Mean delay in the downlink queue for M/G/1 systems is $\overline{W_d} = \frac{\lambda_{id}\overline{T_{dd}^{(2)}}}{2(1-\rho_{dtot})}$, where $\overline{T_{dd}^{(2)}}$ denotes the second moment of the downlink data service time. For the calculation of the delay for uplink traffic, we may view the node as if it had two queues with different priority: the data request queue and the data packet queue, with the former having higher priority. Mean delay may be obtained as $\overline{W_u} = \frac{\lambda_{iu}\overline{T_{ud}^{(2)}}+\lambda_{id}\overline{T_{ur}^{(2)}}}{2(1-\rho_{ur})(1-\rho_{ud}-\rho_{ur})}$.

4 Performance of the Cluster with Bidirectional Traffic

We will now investigate the performance of an 802.15.4 cluster through analytical modeling. We have assumed that the cluster operates in the ISM band at 2.4GHz with raw data rate 250kbps, and with $SO = 0$, $BO = 0$. Furthermore, we have assumed that the minimum value of backoff exponent *macMinBE* is set to three, the maximum value of the backoff exponent *aMaxBE* is set to five, and the maximum number of backoff attempts is set to five, i.e. *macMaxCSMABackoffs* =4.

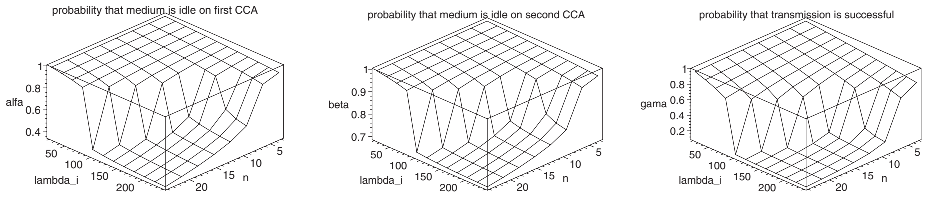
The packet size includes all physical layer and MAC layer headers, and it is expressed as a multiple of unit backoff periods. We also assume that the physical layer header has six bytes and that the MAC layer header and Frame Check Sequence fields have a total of nine bytes. Such a short MAC header implies that the destination addressing mode subfield (bits 10-11) within the frame control field is set to 0 and that the source addressing mode field (bits 14-15) is set to short address mode. This means that packet is directed to the coordinator with the PAN identifier as specified in the source PAN identifier field.



(a) Total offered load on downlink

(b) Offered load at the uplink

Fig. 6. Cluster stability (analytical results)



(a) Probability α. (b) Probability β . (c) Probability γ.

Fig. 7. Probabilities that the medium is idle on first, second CCA and probability of success

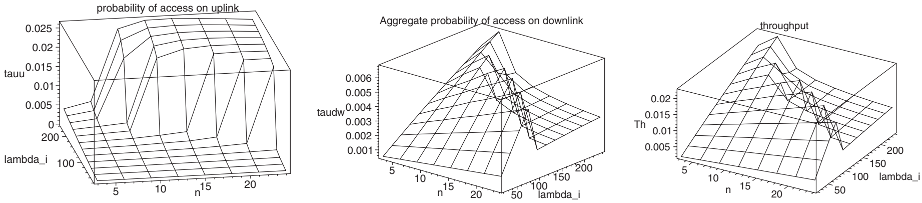
According to the standard, the duration of the MAC command frame for a data request is 16 bytes, but we have rounded it to 20 bytes. In the same manner, the duration of acknowledgment was set to one backoff period, as its duration is 11 bytes.

We consider the scenario where each node sends packets to every other node with equal probability. Therefore, if the uplink packet arrival rate per node is λ_{iu}, then each node receives data at the rate of $\lambda_{id} = \tau_{ud}\gamma$. We have fixed the data packet size to $G'_p(1) = 3$ backoff periods, while the packet arrival rate was varied between 1 arrival per minute to 240 arrivals per minute (4 arrivals per second).

The calculated offered loads for the downlink queue at coordinator and the uplink queues at the node are shown in Fig. 6. We clearly see that downlink offered load is the most critical factor of cluster stability, since it reaches the boundary value of one for moderate network sizes. When this stability condition is exceeded, packet service times and access delays experience large growth. Uplink stability, on the other hand, implies network sizes of less than 25 nodes, with packet arrival rates of at most three to four packets per second.

Fig. 7 shows the probabilities α, β, and γ – i.e., the probability that the medium is idle on the first CCA, the probability that the medium is idle on the second CCA, and the probability of success. We note that α, β and γ reach lower (saturation) bounds at moderate loads for network size between 10 and 20 nodes. The lower bound for the success probability is close to zero, which means that, in this regime, virtually no packet is able to reach its destination.

Fig. 8 shows the uplink and downlink access probabilities, as well as the throughput. The flattening of uplink access probability indicates that the onset of saturation regime, in which case all accesses to the medium are contributed by the request packets that do not succeed. A rather dramatic decrease of downlink access probability for the coordinator may be observed as well; it is caused by the inability of the coordinator to receive any correct data requests due to collisions and blocking. This observation is also confirmed by the diagrams that depict the throughput, which show that the throughput deteriorates rapidly when the cluster enters saturation.



(a) Uplink access probability

(b) Downlink access probability

(c) Cluster throughput

Fig. 8. Pertaining to cluster performance (analytical results)

5 Conclusion

In this work we have modeled the operation of the MAC sublayer of a beacon enabled 802.15.4-compliant network with both downlink and uplink traffic. The model considers acknowledged uplink transmissions and includes the impact of network and traffic parameters such as packet arrival rate, number of stations, packet size, and inactive period between the beacons. We have modeled the interaction between uplink data queues, uplink request queues and downlink data queues and evaluated the stability criteria of those queues. We identify the downlink queue stability as the tightest criterion for the network given the setting where nodes uniformly communicate among themselves. All these results indicate that the network coordinator can handle only a small amount of downlink traffic and that the number of nodes and their traffic load should be chosen with the goal of keeping the operating point of the network well away from the saturation point.

References

1. E. H. Callaway, Jr. *Wireless Sensor Networks, Architecture and Protocols.* Auerbach Publications, Boca Raton, FL, 2004.
2. J. A. Gutiérrez, E. H. Callaway, Jr., and R. L. Barrett, Jr. *Low-Rate Wireless Personal Area Networks.* IEEE Press, New York, NY, 2004.
3. Standard for part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (WPAN). IEEE Std 802.15.4, IEEE, New York, NY, 2003.
4. G. Lu, B. Krishnamachari, and C. Raghavendra. Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks. In *Proc. Workshop on Energy-Efficient Wireless Communications and Networks EWCN'04*, Apr. 2004.
5. J. Mišić, S. Shafi, and V. B. Mišić. Performance of 802.15.4 beacon enabled PAN with uplink transmissions in non-saturation mode - access delay for finite buffers. In *Proc. BroadNets 2004*, pages 416–425, San Jose, CA, Oct. 2004.

Analysis of Windowing and Peering Schemes for Cache Coherency in Mobile Devices

Sandhya Narayan, Julee Pandya, Prasant Mohapatra, and Dipak Ghosal*

Department of Computer Science,
University of California, Davis, CA 95616, USA
ghosal@cs.ucdavis.edu

Abstract. A major factor in determining the effectiveness of caching in wireless networks is the cache coherency scheme which maintains consistency between mobile stations (MSs) and the server. Since the wireless channel is inherently a broadcast medium, an appropriate cache coherency scheme is one in which the server broadcasts cache invalidation reports (IRs) that contain data update information. However, in a wireless environment, since MSs may connect to the network only intermittently (e.g., to save power), IRs may be missed. This would cause the MS's cache to become invalid and in turn the cache would have to be purged resulting in higher query-delay and lower throughput. One approach to improving the cache coherency for mobile devices is the Time Stamp (TS) method [2] which uses a *windowing* scheme. In this scheme, the IR in a particular interval contains the IRs for a number of previous intervals determined by the window size. Another orthogonal approach to improving cache coherency is the *peering* scheme [10] where an MS can query neighboring peers to retrieve IRs that it may have missed while it was disconnected. In this paper, we present a unified mathematical model based on Discrete Markov Models (DMMs) to study the effectiveness of these orthogonal schemes both individually as well as their relative importance when they are implemented together. The results show that both schemes are comparable for the most part. Since they are orthogonal, they can be combined in ways that is tailored for the particular environment to achieve significant improvement in performance.

1 Introduction

For wireless networks and applications, caching data at the clients reduces unnecessary network accesses and use of wireless channels, saving energy as well as wireless bandwidth [9, 13]. Benefits of caching can be maximized through robust and effective placement of data objects, maintaining coherency, replacement algorithms of cached objects, and various timing issues. This paper addresses the

* S. Narayan, P. Mohapatra, and D. Ghosal are with the Department of Computer Science, University of California, Davis. J. Pandya is with the MIT Lincoln Labs.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 240–251, 2005.

problem of maintaining cache coherency specifically in the context of intermittently disconnected devices.

The underlying cache invalidation method studied in this paper is based on periodic server broadcasts of Invalidation Reports (IRs) to the clients. These reports indicate which data items have been modified during the last interval. Clients use the IRs to update their caches. One of the key issues of IR based techniques arises due to the intermittent disconnectedness of the clients. Since connecting to the network uses power and because power is a limited resource, wireless devices may connect to the network only when necessary. During periods when the device is disconnected, IRs will will be missed. As result, upon reconnection, the client will not be able to guarantee that the cache data is valid and will have to purge its cache. This method, known as the TS scheme, was first discussed in the well known paper [2] in which they also proposed an windowing scheme as an enhancement to the basic approach. Under this scheme, each IR in a particular interval contains the IR for a number of previous intervals defined by the window size. The paper provided analysis based on asymptotic bounds and studied the impact of the window size on the cache hit rate and throughput.

An orthogonal approach for improving cache coherency is to exploit peering among MSs [10]. In this approach, called Peer Enhanced Cache (PEC), MSs utilize the ad-hoc mode of operation, which is now available with most wireless interfaces. They store IRs on behalf of other MSs which are disconnected. If an MS misses IRs due to disconnection, it can attempt to retrieve them by querying all the peers within its direct transmission range. Note that PEC is an orthogonal scheme and can be implemented in conjunction with other cache invalidation methods, in particular with the TS scheme.

In this paper we present a unified mathematical model to study the performance of the windowing and peering schemes. The analysis is based on Discrete Markov Model (DMM) [1]. The model allows us to study the windowing and the peering schemes individually and when they are implemented together. Based on a thorough analysis of the system, the following are the main contributions of this paper:

- While the cache consistency using windowing in TS approach was analyzed in [2], the authors only provided upper and lower bounds for the cache hit rate and throughput. In this paper, we provide an accurate analysis of the TS scheme.
- The peering scheme analyzed in [10] modeled only the basic scheme, i.e., with window size of 1. This model extends the study to the case when both peering and windowing are used.
- The results show that both schemes provide significant improvements to cache coherency, and that their performance is comparable for the most part. However, both windowing and peering consume bandwidth and thereby reduce the gain in throughput. Since they are orthogonal, they can be combined in ways that is tailored for the particular environment to achieve significant improvement in performance.

The remainder of this paper is organized as follows. Section 2 presents a reference architecture to describe the peering and windowing schemes. Section 3 outlines the model and presents a mathematical analysis of the windowing and peering schemes and when they are implemented together. Section 4 presents the derivation of some metrics such as hit rate and throughput. Section 5 discusses the results and Section 6 provides a brief description of related work. Finally conclusions are drawn in Section 7.

2 Reference Architectures

An application program running on a Mobile Station (MS) uses a local cache for performance, bandwidth savings, and energy efficiency. An application query is handled by the cache if possible, otherwise, it is sent to the base station (BS) which also acts as a server. The Base Station (BS) responds with the data objects for that query which are then cached by the MS. The BS periodically (every L time units) broadcasts an Invalidation Report (IR), which indicates the data items have been modified during the past interval. All connected MSs within the base station coverage area (BSCA) receive the IR. They use this to validate their cache contents and discard data objects that have been invalidated by the IR. During periods when an MS is disconnected, it does not receive IRs. Therefore, if disconnection times is greater than the IR interval, and one or more IRs is missed, when the MS reconnects, the cached data cannot be guaranteed to be valid and will have to be purged.

2.1 TS Scheme

In the TS scheme (also called broadcast time stamp method) [2], the BS broadcasts IRs every L time units as before. However, each IR indicates the items that have been modified during a specified window of time w, where w is some multiple of L. The larger the w, the longer the client can disconnect without having to purge its cache. However, increasing w also increases the size of the IR which implies higher bandwidth for each broadcast. Note that in all IR based schemes, the MS has wait to answer queries until the next IR is broadcast. Queries generated between broadcasts are queued until the next IR is received. This is done so that data items that became invalid during the IR interval are not retrieved from the cache and used to answer the queries. Therefore, if the IR interval can be decreased, the query latency will also decrease.

2.2 Peer Enhanced Caching (PEC)

An orthogonal scheme to improve the IR-based cache coherency mechanism is PEC which exploits peering between MSs [10]. We denote peer coverage area (PCA) to be the coverage area defined by the transmission range of the peer. In order to assist other peers, each MS stores the IRs it receives when connected to the network. When a MS reconnects after being disconnected, it broadcasts a query in its PCA requesting for the IRs that it has missed. All the MSs that are within the PCA respond with the subset of the requested IRs that they have. The requesting MS then waits until the next IR and based on the IRs received from its peers, it determines if it can validate the cache. If not, the cache is purged. Note that the larger the number of peers, the longer the client can disconnect without having to purge its cache. However, the more the number of peers the higher the bandwidth consumption.

3 A Unified Model

We consider that time is slotted and the slot length is one IR interval denoted by L. The sleep-awake cycle is modeled by a first-order Discrete Markov Model [1] consisting of two states - a sleeping (disconnected) state and 2) an awake (connected) state. When awake, the MS can get disconnected in an interval with probability d. The duration of being in the *awake* state is negative exponentially distributed with rate $1/d$. We assume that the disconnection time is also negative exponentially distributed with a mean value of T_d. Since L is the length of the IR interval, the probability that a sleeping MS would continue to sleep in the next IR is given $s = e^{-L/T_d}$. Similar to the two papers of interest ([2] and [10]), we assume that the probabilities s and d are equal.

The models for the query and update are same as that used in [2]. Each MS queries a subset of the database with a high locality. This subset is known as the hot spot and each item in the subset is queried at rate λ. Updates to each data item are negative exponentially distributed with mean rate μ updates per second. We consider a random mobility model in which at the end of each IR interval, the MS moves to a random location within the BSCA. Under this model, the number of peers within the PCA will be proportional to the ratio of the areas of the PCA and the BSCA. We will let N denote the average number of peers in the PCA.

In addition to the above parameters, we use the following notation adopted from [2]. The probability of being awake and having no queries in an interval is $q_0 = (1-s)e^{-\lambda L}$. The probability of no queries in an interval is $p_0 = s + q_0$. The probability of no updates during an interval is $u_0 = e^{-\mu L}$.

3.1 Analysis of the Windowing Scheme

Let k denote the window size. As in [2], the goal of the analysis is to find the probability that a tagged MS, denoted by MS_t, has slept for k or more consecutive intervals between two consecutive queries that are i intervals apart. Based on the model, the states of MS_t can be modeled by a DMM shown in Figure1. There are $k+1$ states. State 0 indicates that the MS is awake and it has all the required IRs and that its cache is in a consistent state. State m for $0 < m < k$, indicates that the MS has slept for $m-1$ consecutive intervals and that it can still get all the missed IR data if it wakes up at the next interval. State k indicates that MS_t has missed some IR data as a consequence of sleeping for k or more consecutive intervals. The cache has become inconsistent as a result

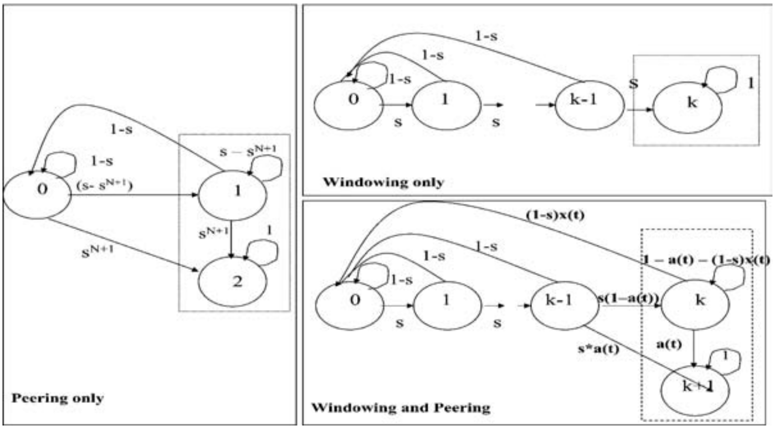


Fig. 1. The Discrete Markov Model for windowing only, peering only, and the combined schemes

and needs to be purged. It should be noted that the system will not remain in this state forever. In fact, it will get back to state 0 as soon as the cache is purged and valid data is retrieved from the server. The diagram also shows the state-transition probabilities.

Let $P_m(t)$ be the probability that the DMM is in state m at time t. Without loss of generality, we assume the following initial conditions: 1) $P_0(t) = 0$ for $t < 0$ and $P_0(0) = 1$ and 2) $P_m(t) = 0$ for $m > 0$ and $t \leq 0$. For $m = 0$, $P_0(t) = \sum_{j=0}^{k-1} P_j(t-1)(1-s)$. Since, $\sum_{j=0}^{k} P_j(t) = 1$, $P_0(t) = [1-P_k(t-1)](1-s)$. For $m = 1, 2, ..k-1$, $P_m(t) = sP_{m-1}(t-1)$ and for $m = k$,

$$P_k(t) = sP_{k-1}(t-1) + P_k(t-1) = s^k P_0(t-k) + P_k(t-1). \tag{1}$$

Substituting for $P_0(t-k)$ we get

$$P_k(t) = P_k(t-1) + s^k(1-s)(1 - P_k(t-1-k)), \tag{2}$$

for $t \geq k$ and $P_k(t) = 0$ for $t \leq k$. This difference equation can be used to compute the probability at time t that a MS has slept for k or more consecutive intervals over a period of t intervals.

3.2 Analysis of the Peering Scheme

Let N denote the number of peer in the PCA. The state transition diagram of the DMM is shown in Figure 1. State 0 indicates that it has all the required IRs and that its cache is in a consistent state. State 1 indicates that the MS has slept for one or more consecutive intervals and missed some IR data. However, it can still get the missed IR from other peers that have it. State 2 indicates that none of the MS has valid IR. The diagram also shows the state-transition probabilities. Note that there is a transition from both State 0 and State 1 to

State 2. This corresponds to the case when MS_t and all its N peers are asleep in the same interval. As a result no MS will have valid IR for that interval.

It can be shown that the probability of having incomplete IR data is given by

$$P_2(t) = s^{N+1} \sum_{k=0}^{t-1} (1 - s^{N+1})^t = 1 - (1 - s^{N+1})^t. \tag{3}$$

Thus the probability of having valid IR at time t is $1 - P_2(t) = (1 - s^{N+1})^{t1}$. The effect of peering is evident from the above equation; the effective sleep probability is reduced from s to s^{N+1}, thereby significantly improving cache performance.

3.3 Analysis of the Combined Scheme

In this case we consider both peering and windowing. As before, k is the window size for the IR and N is the number of peers of MS_t. A DMM to describe this case will have $(N+1)^k$ states. We consider an approximate state machine with $(k+2)$ states shown in Figure 1. State k indicates that the MS_t has slept for k or more consecutive intervals and its local IR is incomplete. However, at least one of the peers has required IR and using which the MS could complete the required list of IRs. State $k+1$ indicates that all the MSs have incomplete IRs and the union of all the IRs list is also incomplete. There is a transition from State k to State 0 since MS_t can obtain all the IRs from other peers even after having slept for k or more consecutive intervals. The term $a(t)$ which indicates no peer-help is available is given as:

$$a(t) = (sP_{k-1}(t-1) + P_k(t-1))^N. \tag{4}$$

The term $x(t)$, which indicates that MS_t can get the IRs it missed from other peers, is given by

$$x(t) = 1 - (1 - (sP_0(t-1) + (1-s)(1 - P_k(t-1) - P_{k+1}(t-1)))^N. \tag{5}$$

Using this model, we can compute the probability that the MS_t has incomplete IRs if it slept for k or more consecutive intervals between two queries that are i intervals apart in a situation where there are N other peers helping the MS_t to maintain valid IR. As before, the probability $P_{k+1}(t)$ can be solved recursively.

4 Performance Metrics

The improvement in cache coherency due to windowing and peering is measured by using cache hit rate and throughput as in [2] and [10]. Note that we only consider the hit rate due to cache coherency; effects of cache size and replacement policies are not considered in this study.

[1] In [10], this probability of obtaining valid data at a particular interval t is given as $(1 - s^N)^t$. The derivation had ignored the case where the MS_t also contributes through peering. That is, the IR is valid because MS_t is awake and has a valid IR while all other N peers are asleep. The derivation here accounts for this case.

4.1 Hit Rate

To determine the cache hit rate, we consider two consecutive queries that are i intervals apart. Using the notations defined earlier, the following conditions must be satisfied for a cache hit for the second query: 1) probability that there are no updates during the i intervals which is given by ${u_0}^i$, 2) probability that there are no queries during the $i-1$ intervals between queries which is equal to ${p_0}^{i-1}$, 3) probability that MS_t has a valid IR at interval i which is equal to $P_v(i) = 1 - P_{k+1}(i-1) - (1-x(i-1))P_k(i-1)$, and 4) the first query arrived at interval 1. Taking all these four conditions, the cache hit probability is given by

$$h_{gen} = (1-p_0)\sum_{i=1}^{\infty} {u_0}^i {p_0}^{i-1} P_v(i). \tag{6}$$

For the case of $k = 1$, the term $P_v(i) = 1 - P(2) = (1 - s^{N+1})^i$. For the case of $N = 0$, the term $P_v(i) = 1 - P_k(i-1)$, as state $k+1$ is renamed as k.

4.2 Throughput

To compute the throughput we follow the analysis done in [2] and [10]. As mentioned before, we assume that window w is a multiple of L and thus $w = kL$ and the bandwidth of the wireless network is W. Therefore, bandwidth available for each interval is LW. We assume that the number of bits per up-link query is b_q, and the number of bits per query answer is b_a. Each timestamp takes b_T bits and there are n items in the database.

Without Peering. The bandwidth available for queries is $LW - B_c$, where B_c is the bandwidth needed to broadcast the IR and is computed as follows: The expected number of items that changed in window w denoted by $nc = n(1 - e^{-\mu kL})$. Total size of the IR denoted by $B_c = b_T + n_c(log(n) + b_T)$, where $log(n)$ is the number of bits needed to transmit an object's ID, and b_T bits are needed to transmit its timestamp. The first b_T term represents the bits needed to send the IR report's timestamp.

Throughput T is the number of queries processed per interval by MS_t. Throughput due to cache misses $= T(1-h)$, where h is the cache hit rate. Traffic due to queries that resulted in cache misses $= T(1-h)(b_q + b_a) = LW - B_c$ from which we get

$$T = (LW - B_c)/(1-h)(b_q + b_a). \tag{7}$$

We can substitute for the hit rate derived in the previous section and compute the throughput.

With Peering. In this case the bandwidth available for queries will be lower because the communication between the peers uses up some of the bandwidth. Let B_{P2P} be the bandwidth used for communication between peers which consists of two components, i.e., $B_{P2P} = B_{P2P1} + B_{P2P2}$ where B_{P2P1} is the bandwidth required to query other other peers and B_{P2P2} is the bandwidth required to respond to a peer query.

Calculating B_{P2P1}. Let b_{req} be the fixed number of bits required to broadcast request to peers. Let N_{res1} be the number of peers that respond and b_{res} be the size of each response. Let p_{reqh} denote the probability that MS_t will send out query to its peers and let p_{resh} denote the probability that a peer receiving the query will respond with a subset of of the IR requested by MS_t. Then

$$B_{P2P1} = p_{reqh} * (b_{req} + N_{res1} * b_{res}). \tag{8}$$

where $p_{reqh} = (1-s)s^k$. Further, $N_{res1} = N * p_{resh}$ where $p_{resh} = (1-s)(1-s^k)$, as it must be awake and has not slept for k consecutive intervals. Thus,

$$N_{res1} = N * (1-s)(1-s^k). \tag{9}$$

The expected number of intervals asleep is equal to $s/(1-s)$ as derived in [10]. Therefore, conditional expectation of number of IRs needed given that the MS_t has slept for k or more intervals is still $s/(1-s)$, although the probability of this occurring is much smaller (s^k). Thus the size of this response $b_{res} = s/(1-s)*B_c$, where B_c is the mean bandwidth needed to transmit an IR.

Calculating B_{P2P2}. Let b_{req} be the bits used to broadcast request to peers. This has a fixed and known value. Let N_{req2} be the number of peers that request for peer help and N_{res2} be the number of responses sent by MS_t. As before, let b_{res} be the size of each response. Finally, let p_{vIR} denote probability that the MS receiving the query has at least one valid IR. Let B_{P2P2} denote the bandwidth required to respond to a query and is given by

$$B_{P2P2} = (1-s)(N_{req2} * b_{req} + (p_{vIR}) * N_{res2} * b_{res}) \tag{10}$$

where, $p_{vIR} = (1-s^k)$, $N_{req2} = N * Pr(MS\ requesting\ help)$ which is equal to $N * ((1-s) * s^k)$. Finally, $N_{res2} = N_{req2}$ because the MS_t is assumed to respond to all requesters, if it has valid IRs.

5 Results and Discussions

All the experiments were run with the following parameter values: the length of the IR interval L was set to 10 seconds, two different query rates were used, i.e., λ was set to 0.001 or 0.1 queries per second, and the update rate μ was set to 0.0001 updates per second.

5.1 Peering Only

The hit rate and improvement to throughput in the peering only scheme are shown in Figures 2. As expected the hit rate decreases with increasing s. Also the hit rate increases with increasing query rate λ. This is because when the query rate is high, queries occur at fewer intervals apart, where the probability of having incomplete IR data is lower. When queries occur far apart, the hit rate is lower as the probability of having incomplete IR data is higher. Therefore, at

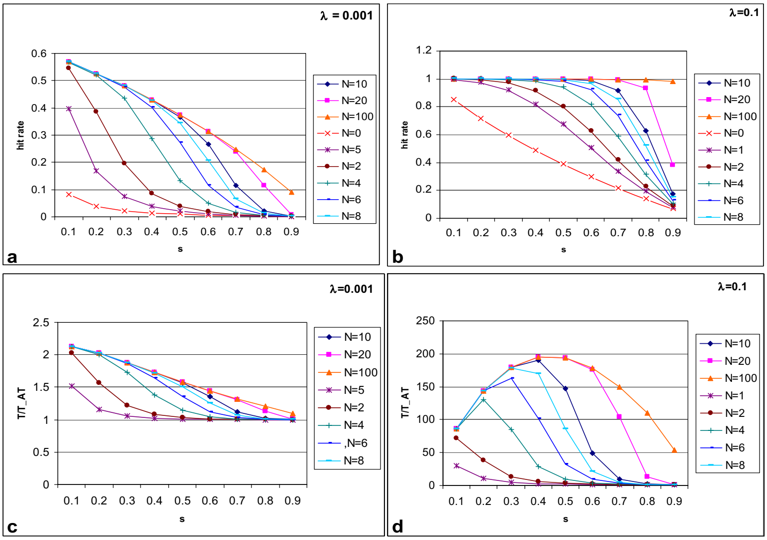


Fig. 2. Hit rate (a and b) and throughput (c and d) for peering only scheme (Note that normalizing throughput T_{AT} is for $k = 1$ and no peering)

a low query rate ($\lambda = 0.001$), the hit rate is very low. As queries come farther and farther apart, the probability of having incomplete IR data will eventually be 1 (for any value of s except 0).

The hit rate increases with increasing N. The benefit of having N peers is to reduce the effective sleep probability from s to s^N (see Equation(3)).

As expected, the throughput improvement is much higher for high query rate ($\lambda = 0.1$) as compared to low query rate ($\lambda = 0.001$). The benefit of peering levels off at higher values of s because the increased peer-to-peer traffic consumes bandwidth.

5.2 Windowing Only

The results of windowing scheme is shown in Figures 3. For high query rates, the benefit of windowing is more at higher values of s. This is because when ss is high, the hit rate is low for the case without windowing. Windowing reduces the effective sleep probability and thus increases the hit rate. For low query rates, the effect of windowing is more significant at lower values of s. At higher values of s, windowing is not as effective because the queries are too far apart for any windowing to make a difference. With respect to the throughput, as in the case of peering, the improvement drops at higher values of s because of higher bandwidth overhead due to larger window size.

5.3 Combined Scheme

The improvement to hit rate and throughput due to a combination of windowing and peering is shown in Figures 4. To study the relative effectiveness of window-

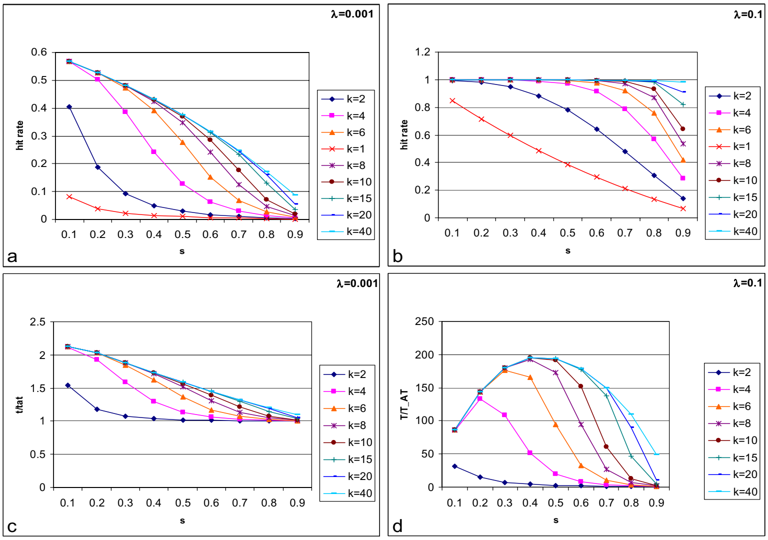


Fig. 3. Hit rate (a and b) and throughput (c and d) for windowing only scheme (Definition of T_{AT} same as before)

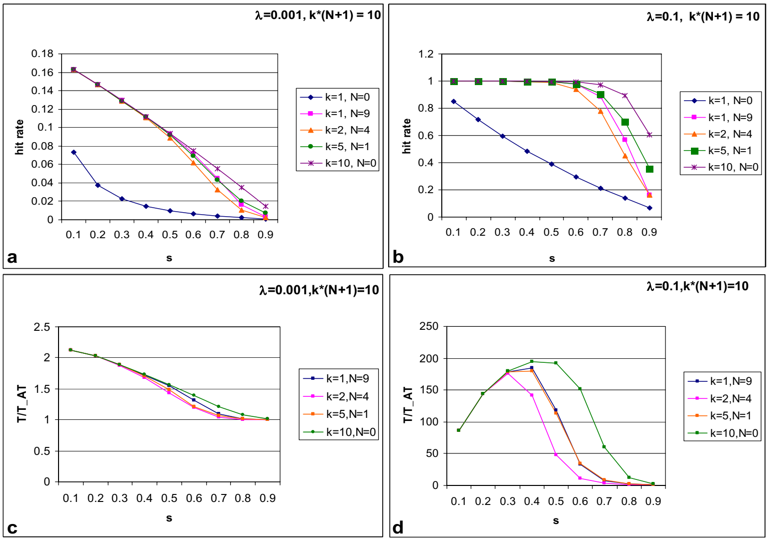


Fig. 4. Hit rate (a and b) and throughput (c and d) for the combined (Definition of T_{AT} same as before)

ing versus peering, we chose N and k values such that the product $(N+1)*k$ is constant denoted by C and set equal to 10 for results shown here. For high query rates increasing k improves the hit rate better. When $k = C$ (i.e., only windowing), the probability of having incomplete IR data for the first $(C-1)$ intervals is equal to zero and is equal to s^C at interval C. In contrast, when $N+1=C$ (only peering), the conditional probability of having incomplete IR data is s^{N+1} in each of the C intervals and the probability of having incomplete IR data is $1-(1-s^C)^C$ at interval k. When query rate λ is high, the first few intervals are more important in the analysis, as the second query is expected in that duration. However, during this same period, as shown above, probability of having incomplete IR data is higher in the case of peering. Therefore, the case for windowing performs better than peering.

6 Related Work

In this paper we studied two methods to improve cache invalidation upon update using IRs ([2], [10]). Many other works ([5], [4], [11], [3], [12], and [8]) also dealing with cache invalidation have been published. Group-based invalidation schemes were proposed in [8] to retain as many valid objects as possible by checking for cache validity with the server upon reconnection. A different type of IR was proposed in [6] where the IR contains a set of bit sequences, each associated with a timestamp. The bits represent data objects in the database and indicate change to objects. The scheme reduces the number of cached objects discarded, but increases the size of the IR report. An asynchronous and stateful approach was proposed in [7] to reduce query latency and number of discarded objects. In [5] the authors have proposed an adaptive algorithm which combines the TS and Bit-Sequence approaches based on current query and update rate to achieve better performance. Another approach to reduce query latency and improve bandwidth utilization was proposed in [4] by repeating a small fraction of the IR information within the IR interval. In [11] the authors evaluate the cache performance when using some of the above schemes.

7 Conclusions

In this paper we proposed a unified mathematical model for studying the improvement to cache coherency using windowing, peering and a combination of both. From the results we conclude that both methods provide significant improvements to cache performance, and that their performance is comparable for most part. Both windowing and peering consume bandwidth and thereby reduce throughput. The extra bandwidth consumed by windowing is of the order of k, while for peering it is of the order of N^2. Since the two schemes are orthogonal, they can be combined to improve cache performance greatly. Depending on the relative costs of bandwidth and power consumption for Base Station to Mobile Station and peer-to-peer communications, one can choose to combine the two methods in a way to optimize the cache performance. Future work can include studying the performance of the three schemes using different cost models for bandwidth and power consumption.

References

1. A.W.Drake. Discrete state markov processes. *Chapter 5: In Fundamentals of Applied Probability Theory*, 1967.
2. D. Barbará and T. Imieliński. Sleepers and workaholics: caching strategies in mobile environments. *MOBIDATA: An Interactive journal of mobile computing*, 1(1):1–12, 1994.
3. J. Cai and K. Tan. Energy-efficient selective cache invalidation. *Wireless Networks*, 5(6):489–502, 1999.
4. G. Cao. A scalable low-latency cache invalidation strategy for mobile environments. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 200–209. ACM Press, 2000.
5. Q. Hu and D. K. Lee. Cache algorithms based on adaptive invalidation reports for mobile environments. *Cluster Computing*, 1(1), 1998.
6. J. Jing, A. Elmargarmid, S. Helal, and R. Alonso. Bit-sequences: An adaptive cache invalidation method in mobile client/server environments. *ACM/Baltzer Mobile Networks and Applications*, 2(2), 1997.
7. A. Kahol, S. Khurana, S. Gupta, and P. Srimani. A strategy to manage cache consistency in a distributed mobile wireless environment, 2000.
8. P.S.Yu K.L.Wu and M.S.Chen. Energy-efficient caching for wireless mobile computing. In *Proceedings of the 12th International Conference on Data Engineering*, pages 336–343. IEEE, February 1996.
9. P. Nuggehalli, V. Srinivasan, and C. Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *The Fourth ACM International Symposium on Mobile Ad Hoc Networking & Computing MOBIHOC 2003*, 2003.
10. J. P. Pandya, P. Mohapatra, and D. Ghosal. Asymptotic analysis of a peer enhanced cache invalidation scheme. In *Proceeding WiOPT: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 200–209. IEEE Press, 2004.
11. K. Tan, J. Cai, and B. Ooi. An evaluation of cache invalidation strategies in wireless environments. *IEEE Transactions on Parallel and Distributed Systems*, 12(8):789–807, 2001.
12. B. Zheng, J. Xu, and D. Lee. Cache invalidation and replacement strategies for location-dependent data in mobile environements. *IEEE Transsactions on Computers*, 51(10):1141–1153, 2002.
13. R. Zheng, J. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing MOBIHOC 2003*, 2003.

Fast Scalable Robust Node Enumeration

Richard Black[1], Austin Donnelly[1], Alexandru Gavrilescu[2], and Dave Thaler[2]

[1] Micrsoft Research Ltd., 7 J.J. Thomson Avenue, Cambridge, CB3 0FB, UK
[2] Microsoft Corp., One Microsoft Way, Redmond, WA 98052, USA
{rjblack, austind, alexang, dthaler}@microsoft.com

Abstract. In a Local Area Network of computers, often a machine wants to learn of the existence of all the others satisfying some condition. Specifically, there are a number of existing discovery algorithms which permit an enumerator to reliably discover protocol participants, many of them idealised. This paper provides a new technique which controls the load placed on the network, minimises the time to completion, handles networks with significant loss, and scales over many orders of magnitude. Most significantly, the protocol also deals with the possibility of a malicious enumerator; an important contribution needed for current real-world networks. We also address the effects of several systems and engineering aspects, including scheduler jitter and clock quantisation.

1 Introduction

In a Local Area Network of computers, often a machine wants to learn of the existence of all the others satisfying some condition (such as being prepared to co-operate to perform a task). There are a number of existing discovery algorithms which permit an enumerator to reliably discover protocol participants. For example, various service discovery protocols (such as SSDP [3]) fall into this category, as do node discovery protocols (such as Browser [8]).

Some discovery protocols such as IGMP [2] do not wish to enumerate all the participants, but only wish to know whether there is at least one participant satisfying some property present. We do not consider this class of problem in this paper.

Closer to our problem is the case of link-local broadcast or multicast pings; they differ in that that they do not include reliability (though some of our techniques could perhaps be applied to that domain).

In this paper we consider the application in which enumeration of participants is initiated by a single *enumerator* broadcasting a Request message. In response to this, *responders* send a Response message, thus revealing their presence to the enumerator.

To avoid an implosion of traffic at the enumerator, a scalable enumeration mechanism requires a scheduling method whose purpose is to decide *when* the responder should send its Response. For example, the most common scalability method is a simple random delay within a fixed time interval.

We require a solution which also provides reliability (with high probability all nodes are enumerated), promptness (a small network takes a shorter time than a large network) and defence against a malicious enumerator. We say more about our requirements in

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 252–263, 2005.

section 3.1, and comparison with previous work in section 2, but first we state more clearly what we mean by a malicious enumerator.

A malicious node on a local area network can engage in a number of extremely disruptive actions. It can consume all the bandwidth and prevent normal communication. It can also fake its source address in packets so as to reroute ("steal") traffic for another node. Given the possibility of such villainy it may seem pointless to discuss the operation of a protocol in the presence of malicious nodes. What we are attempting to defend against, however, is the incorrect apportionment of blame for bad behaviour on the network. Specifically, we wish to prevent a bad node from using a small amount of bandwidth to trigger good nodes to use a large amount of bandwidth. Our requirement is a protocol which cannot be tricked into causing the aggregate traffic load of good nodes to exceed the specified target rate.

In this paper we cover related work in section 2. In section 3 we introduce two load-adaptive algorithms which would be expected to have similar idealised behaviour: one which keeps a fixed probability of transmission and varies the periods over which it adapts to the load on the network, and one which operates on a fixed timescale but varies the probability. We evaluate these by analysis and simulation, showing that they behave differently in practice. Finally we recommend the preferred algorithm.

2 Related Work

There have been many examples of distributed load control for Media Access since randomised distributed media access was first implemented in 1970 in the Aloha system [1]. The most famous and commonly used is probably that in Ethernet, binary exponential back-off. Our work is dissimilar to controlling the load at the Media Access level; a MAC protocol is designed to arbitrate a single segment at full line rate, not a LAN comprised of many segments and where we want to consume only a small fraction of the available resource.

In work on scalable address allocation [5] and scalable reliable sessions [7], the problem under consideration was ensuring, regardless of whether there were a small or large number of potential responders, that a reasonable number (neither too large nor too small) responded to the request. Again our work is dissimilar in that the intention is that all responders should eventually respond, but at a rate which is below the rate directly achievable from the media in use.

Other work considers the problem of having agents in a network gathering and sharing information about each other in order to increase their knowledge of the network. Most packet routeing techniques work like this, either at a network layer directly, or at an overlay layer. There have been hundreds of publications in the field of *ad-hoc* sensor wireless networks addressing that problem which we do not attempt to list.

Our work is different in that all the agents are expected to be attached to substantially the same network, and only the enumerator is attempting to gather the information. In particular we address the case where one node's communication may interfere with another's and so the aggregate load must be controlled.

A specific example of more closely related work is resource discovery [6]. However, that work addresses the case where there is no centralised coordination point. In our work we address the problem where there is a centralised coordination point, but the responders do not trust it to provide correct load control on the network.

Many other discovery systems (e.g., [6, 8]) assume that one node can be trusted to pass on the discovery of other nodes. Once again, our work is different in that the problem we address precludes this because (a) each node may have to include information with its response which makes responses too large to combine and (b) nodes may not trust each other.

Finally, Scalable Timers and RTCP [9, 4] have some similarities in that overall transmission of information is used to control each nodes transmission of information; however in those works each node sends many packets and the goal is a long term stability. In the enumeration problem each node is sending only once (unless retransmission is required) and is not attempting to find a long-term rate.

3 Enumeration Algorithms

In this section we give more detail on the requirements on our protocol. We then describe an idealised algorithm to use as a performance goal; it is impractical because it requires an oracle to be consulted to determine scaling information. Subsequently we describe our two practical algorithms which we will compare in section 4.

3.1 Requirements

To provide reliability, the enumerator must repeatedly retransmit its Requests. We do not wish responders to send a response to each request therefore the enumerator acknowledges Responses and responders only re-respond on receipt of a request without an acknowledgement. For efficiency, we assume that a large number of acknowledgements can be piggybacked with each Request by listing the responders' addresses in the enumerator's Request packet. Such a behaviour is illustrated in figure 1 where requests are being sent with a fixed interval T_E of 200 ms.

The corresponding behaviour of each Responder is illustrated in figure 2. The first request causes it to move to a pausing state; after some time it will send its Response and move to the sent state. In the sent state an acknowledgement causes it to transition to the done state, whereas a negative acknowledgement (a Request in which it is not acknowledged) causes it to return to the pausing state. A positive acknowledgement also causes a transition from the pausing state to the done state; this transition can occur when a response and a negative acknowledgement cross due to concurrency, followed by a positive acknowledgement.

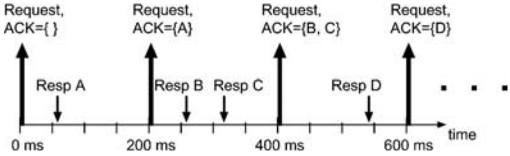


Fig. 1. Enumerator sends regular Requests, acknowledging Responses received in prior period

To provide timeliness, we desire that the enumeration complete as quickly as possible consistent with the desired network load (i.e., we do not want to statically limit the enumeration to proceeding at the slowest rate that might be needed for the largest possible network). Since the enumerator may be malicious we cannot (as many previous techniques do) trust it to control the rate at which responses are sent; hence one feature of the scheduling methods we present is that responders independently measure the load on the network due to the enumeration and use this to time their Response transmissions.

Specifically, we require that Responses are sent in such a way that they can be seen by all the Responders in the system and that by counting such Responses, Responders can execute a distributed load control function. In our descriptions below we describe the methods as using broadcast (or multicast) so that the responses can be counted, however variations are possible in which responders broadcast with some probability z and send directed (uni-cast) responses with probability $(1 - z)$. Provided z is not too small (there are still a statistically valid number of responses seen), it is still possible to estimate the overall number of responses by dividing the observed number by z. Such a change does not reduce the number of packets sent on the network, but it does reduce the number of broadcasts on the network which may be favourable. A potential disadvantage of such a change is that it implicitly assumes that the loss probability of broadcasts and unicasts is the same, which may not be true on wireless networks.

The enumerator stops transmitting Requests once there have been no Responses for long enough that there can be no more responders waiting to respond. Obviously any practical enumerator must defend against a responder which never ceases to transmit Responses, but that does not affect load control, and is not the aspect of malicious behaviour which we are considering in this paper.

Note that one cannot assume that retransmissions are initiated smoothly during operation; a malicious enumerator can arrange to withhold Request messages for a long time and then send a non-acknowledging Request which would cause many responders to be reactivated simultaneously.

Finally, there are several engineering requirements which are necessary for a practical implementation. First, the state retained by each responder must be strictly limited: it is impractical for them to keep a record of the address of the sender of every response they have seen, only the number of such responses seen. Second, we do not permit any significant computation other than increasing a counter when a response packet is received; computation is permitted only when a request packet is seen, or on a timer.

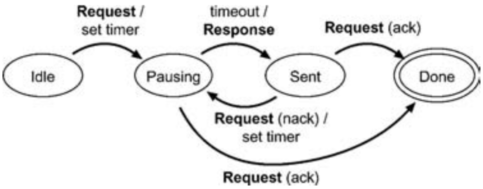


Fig. 2. A responder's state transitions

3.2 Notation

Assume M Response packets are each sent independently at a time chosen randomly and uniformly from an interval T, such that the average load per unit time over the whole interval is $\alpha = M/T$. Then the probability that the number of packets received in unit time, denoted X, has a value k can be approximated by the Poisson distribution:

$$P(X = k) = \frac{\alpha^k}{k!} \cdot e^{-\alpha}$$

For example, if $\alpha = 1$ the probability that a period of unit time has more than three packets is less than 2%, and the probability that it has more than four packets is about one third of one percent. This evaluation of α gives some degree of confidence that such random transmission rarely leads to bursts. We let $I = 1/\alpha$ be the average inter-Response time; hence $T = MI$.

Using q to represent the independent probability of loss on the network, the probability of a Response and a corresponding Acknowledgement both being received successfully is $(1 - q)^2$. The probability p of failure is thus $p = 1 - (1 - q)^2$.

Analytically, the number of Response / Acknowledgement exchanges required using a loss probability q (and combined Response / Acknowledgement loss p) until one of them is successful can be calculated. The expected number of exchanges for a single node is given by the mean of a geometric distribution with probability $(1-p)$ of success, which is $1/(1 - p) = 1/(1 - q)^2$.

We use N to represent the number of responders on the network; and N_{max} to represent the design maximum value for N. The expected number of exchanges for N nodes is thus $N/(1 - q)^2$ over a time $NI/(1 - q)^2$.

3.3 A Best-Case Bound Using Perfect Knowledge

We provide an approximation of a best-case bound on any practical algorithm by considering an impractical algorithm in which the enumerator consults an oracle to obtain the true number of responders on the network, and the responders trust that information.

The enumerator sends out the number of responders, N, in the initial Request message. Each responder schedules a Response at a random time distributed uniformly from the first round which is between 0 and $T_1 = NI$.

In the event of a negative acknowledgement (caused by loss) a responder does not retransmit until the next round (since other responders are still using the current round). Each Request message contains the number of responders still to be seen, hence a responder estimates the loss rate q on the network by comparing this with the initial value of N. As mention in section 3.2 above, if a similar loss rate applied to the Acknowledgements then the number of responders that will still be active can be estimated each round by multiplying N by $(1 - (1 - q)^2)$. This new value becomes the length of the next round in which to uniformly send a linear response.

This impractical algorithm minimises the amount of time needed to enumerate N nodes while maintaining the average load.

3.4 Successive Linear

This practical enumeration algorithm proceeds as a number of loosely-synchronised rounds. The ith round has a duration T_i, which is calculated from the result of the previous round. The first round has constant duration T_1, defined below. At the start of each round, every responder selects a random number: with probability ϕ it sends a Response (at a time chosen randomly, spread uniformly over the round's duration) during the round.

During a round, responders count Responses seen from other responders during the round; suppose the number seen by the end of the round is r_i. The expected value of r_i is given by

$$\mathbb{E}[r_i] = \phi \cdot R_i,$$

where R_i is the actual number of responders that were yet to send at the start of round i. Using r_i as the estimate $\mathbb{E}[r_i]$, yields an estimate for R_i. Of these R_i, approximately r_i have already transmitted, so the estimate of the number remaining at the start of the next round N_{i+1} is given by:

$$N_{i+1} = R_i - r_i \approx \frac{r_i}{\phi} - r_i$$

The responder needs to calculate T_{i+1}, the duration of the next round, so that it can either schedule its Response appropriately or wait the round out. We know that in this next round, on average ϕN_{i+1} responders will transmit a Response, therefore to space them out with average time interval I, we use a round duration of:

$$T_{i+1} = \phi N_{i+1} I = I\phi \cdot (\frac{r_i}{\phi} - r_i) = I r_i (1 - \phi).$$

As each round progresses, r_i decreases and so too does T_i. When r_i gets below a threshold r_{min}, we decide there are sufficiently few responders on the network that further recursive subdivision is pointless, and that the remaining responders should send a Response at times uniformly distributed between now and $N_{i+1}I$.

The constant T_1 (time for the initial round) is calculated based on the worst case of $N_{\max}$ responders on the network. In the first round $\phi N_{\max}$ responders will respond, so to meet the desired minimum inter-Response time of I, we need to spread those responses over time $T_1 = \phi N_{\max} I$.

In the Sent state a responder continues to count Responses. When an unacknowledging Request message causes the responder to move back into the Paused state; its estimate of r_i is therefore the number of packets received since the start of the round in which it sent its response.

Suppose a malicious enumerator sends a negative acknowledgment after xI. For any nacked responder, the round in which it transmitted is ended either prematurely or extended. If it is extended then its r_i count will be larger than ϕR_i and so it will not overload the network. If it is ended prematurely then the x nodes will each have an r_i value of approximately x; thinking that there are $x(1/\phi - 1)$ nodes on the network they will each transmit with probability ϕ over their next interval giving rise to an additional relative load of $x\phi/(x(1/\phi - 1)) = \phi^2/(1 - \phi)$. This is not dependent on x, for small ϕ is small, and in practice the different periods caused by clock jitter make even this effect disappear. Successive Linear is therefore robust to a malicious enumerator.

3.5 Block Adjust

This practical enumeration algorithm is similar to Successive Linear except that instead of fixing ϕ and varying T_i we fix the time over which sampling is performed to a length T_b (the "block time"), and adjust the probability of transmission based on the number of responses seen in a previous round.

At the start of round i each responder has estimated that there are N_i responders left to transmit their responses. Every responder samples its random number generator and chooses a time uniformly distributed between zero and $N_i I$. If this is less than T_b then the responder sends its packet at the chosen time. If the time is greater than T_b then the responder does not send a packet in this round, but counts the number of responses seen during the interval T_b of the round, r_i and uses it to estimate N_{i+1} as follows.

If R_i is the true number at the start of the round then the expected value of r_i is:

$$\mathbb{E}[r_i] = \frac{T_b}{N_i I} \cdot R_i.$$

As before we can use r_i as the estimate $\mathbb{E}[r_i]$ and hence estimate R_i. Of these R_i, approximately r_i have already transmitted, so the estimate of the number N_{i+1} remaining at the start of the next round is given by:

$$N_{i+1} = \frac{r_i N_i I}{T_b} - r_i$$

Eventually some N_i will cause $N_i I \leq T_b$ at which point the responder will transmit in the current round and is finished. We also set N_1 to $N_{\max}$ to set the initial load.

Since a retransmission of a Response packet may be required, each responder continues network loading estimates until it has been acknowledged. In this way, the load due to retransmissions by other responders is continuously being monitored. Thus a value for N_i is always available, should a Negative Acknowledgement cause the Responder to return to the Pending state.

Recall that a malicious enumerator can negatively acknowledge a large number of responders simultaneously. This could cause a load of N/N_i in the extreme case so we deal with this by increasing the estimate of the number of responders by the worst case number which can have become unacknowledged by a Request. Specifically, every time a Request message arrives the number of Response messages received in total (since the Responder left the idle state) is sampled and stored; call this value N_{mb}. At the end of round i the current value of N_{mb} is compared against the value of N_{mb} at the end of round $i-1$ (this value is stored in a further variable ${}_pN_{mb}$). If the value is greater ($N_{mb} > {}_pN_{mb}$) then the difference (equal to the worst case number of Responders that can have moved from Sent state to Pausing state due to Request messages in that round) is therefore added to N_i. Note that ${}_pN_{mb}$ is set on round boundaries and not when a Request message arrives. This adjustment value is an over-estimate for two reasons: first, the enumerator may have positively acknowledged some responders; second, some of the retransmissions from those responders may already have occurred in the previous round.

Several engineering changes are made to the Block Adjust method to take account of real life situations. First, to take account of the fact that jitter is not zero biased, the logic attempts to measure the actual duration T_a of the round (though we do not assume that this has perfect accuracy). Secondly, even using T_a, hosts that are unlucky enough to get many successive large jitters may increasingly over-estimate the number of responders on the network. To stop any possibility of this getting out of hand the logic limits N_i to $100N_{\max}$. Thirdly, we are concerned about the effects of badly written device drivers which may cause packets to be dropped for a significant fraction of a block. Therefore we limit the reduction in the estimated number of hosts in any one round to a factor of three (approximately half an order of magnitude). The final estimator for N_{i+1} is:

$$N_{i+1} = \max\left(\frac{N_i}{3}, \min\left(100N_{\max}, \frac{r_i N_i I}{T_a} - r_i + (N_{mb} - {}_pN_{mb})\right)\right)$$

4 Results and Evaluation

We give an evaluation in this paper using a simulator written specifically for the purpose. This permits us to evaluate the technique under controlled loss, and jitter, and for networks much larger than can be constructed in the laboratory, in addition to comparing with the impractical technique.

The constants chosen for all the methods are designed to give an average load, α, of 1 packet per millisecond (thus $I = 1$ms). This makes them more easily comparable. We set T_E to be 200 ms, $N_{\max}$ to 10000, $r_{\min}$ to 4, T_b to 100 ms and ϕ to 0.1. We also assumed that time intervals can be measured accurately to a resolution of 20 ms. These are compatible with real world values.

It is unrealistic to expect perfect synchronisation, so for some experiments we introduce an extra delay to the time at which responders request to be woken, e.g., to transmit a Response or at the end of a round. This jitter parameter (which is applied randomly and uniformly) is intended to model sender OS-induced wakeup uncertainty together with variations in queueing delays from network and receiver OS; it is always additive (i.e., not zero-biased) because these effects can only ever delay an event, never result in it occurring early.

For loss we simulate receiver loss because that is pessimal for all the calculations (worse than realistic loss): a lost packet contributes to network load, but does not contribute to either (a) progress or (b) any load reducing control mechanism.

Several key aspects of the differences between the algorithms can be seen in figure 3 which shows the network load (averaged over 50 ms buckets and ten simulations) for the three algorithms with 3000 hosts, 100 ms of jitter, and 10% of loss.

- The Successive Linear algorithm uses well below the permitted load during the first 1000 ms due to its inability to adjust its behaviour before the end of its first period.
- The block adjust algorithm, during its first period T_b, transmits at the same rate (appropriate for a network of size $N_{\max}$) as Successive Linear, but rapidly achieves target load.
- Even the impractical algorithm takes a short time to achieve the target load because of the effect of jitter and loss on the network (e.g., because about 10% of the hosts do not see the initial request packet).
- The impractical algorithm has a dip in its usage of the network after 3000 ms; this is the effect of the jitter on its switch from the first phase (in which every node sends

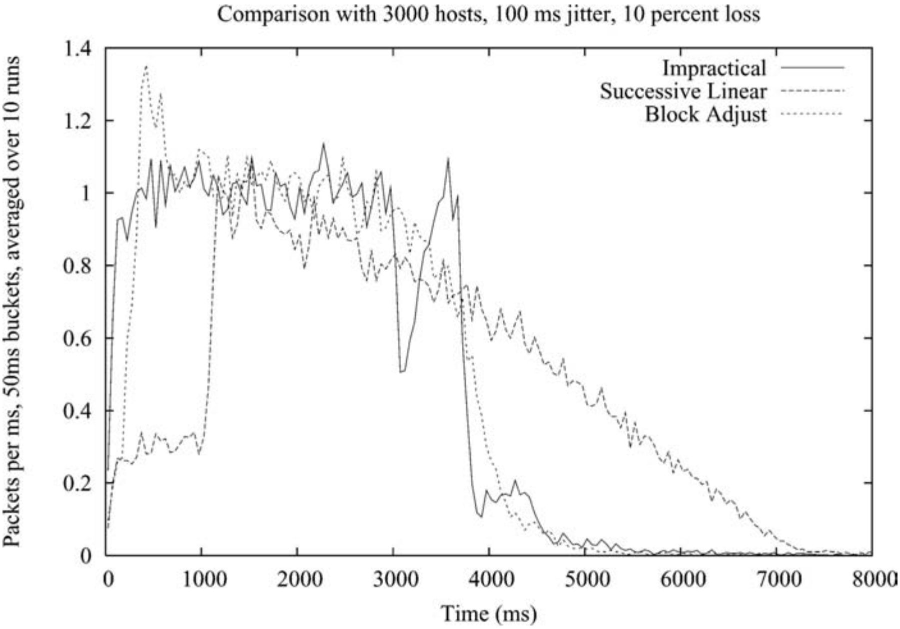


Fig. 3. Network load for 3000 hosts showing differences between algorithms

exactly once) to the second phase in which unacknowledged nodes retransmit after they believe the first phase to be over.

- Although Successive Linear briefly achieves the target load it has difficulty maintaining it and has a long tail. Since the jitter is not zero biased it effectively increases the sampling period; since the sampling period gets shorter in each round, the error increases in relative magnitude with the result that the method increasingly overestimates the load that is actually present.
- Using $N = 3000$, $q = 0.1$, and $I = 1\,\text{ms}$ in our earlier analytical result (without jitter), we get a completion time of 3703 ms, which is consistent with our simulated results here (which do include jitter).
- The block adjust algorithm has a small overshoot near to the start. This is an artefact caused by the large jitter. Since transmission times are delayed by up to 100 ms, the load during any particular period T_b is significantly affected by delayed transmissions from a previous period. This causes the under-damping; however as can be seen the method rapidly stabilises.

We also compared the methods at every half order of magnitude from 1 Responder to the design goal of $N_{\text{max}} = 10000$ responders. To get an idea of how fragile the methods are we also tested them at half an order of magnitude more (30,000) responders; as can be expected the network load in the first period (T_1 for Successive Linear, and T_b for Block Adjust) was approximately three times the target, but subsequently the load returned to target.

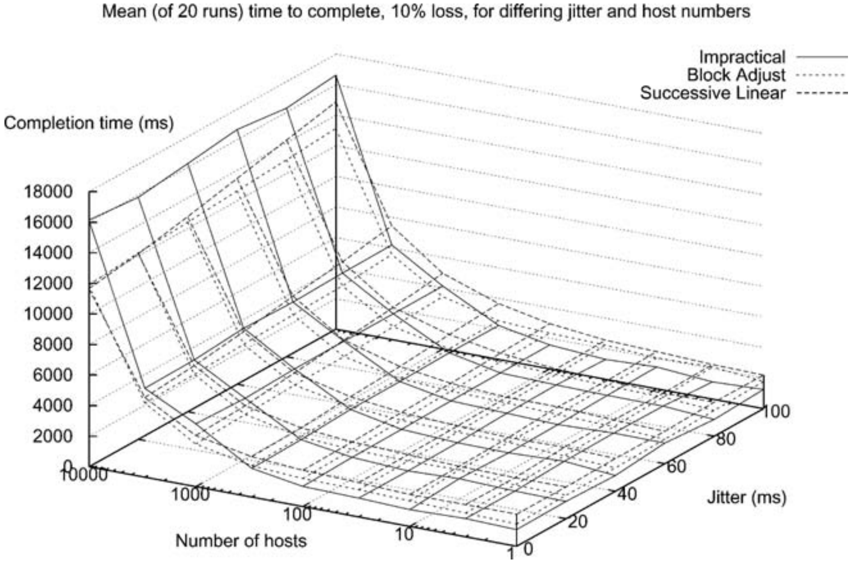


Fig. 4. Completion times

4.1 Completion Time

Figure 4 shows the finishing time for these methods (compared against the Impractical method) both for increasing numbers of hosts, and for increasing amounts of jitter.

It can be seen that with higher number of responders the finishing times become similar and dominated by NI. For fewer nodes, however, Block Adjust is superior because it can adapt more quickly; Successive Linear cannot begin to adjust until the end of the first time period at $\phi N_{\max} I$.

With higher jitter all methods take longer to complete because jitter is not zero biased. Nevertheless it can be seen that Block Adjust is less negatively impacted than Successive Linear.

The effect of loss on these methods can be seen by looking at the baseline data in figure 5. With higher levels of loss all methods take longer because retransmissions are required. Block Adjust retains its advantage over Successive Linear as loss increases.

4.2 Improvements

We originally described acknowledgements as being sent in the subsequent request packet. It is possible, however, to use any spare space within a Request packet to re-acknowledge previously acknowledged responders. The effects can be seen in table 1 by comparing the columns marked “repeat” with the original base-line (marked “none”). Since responders whose responses are received at the enumerator but where the (first) acknowledgement is lost now have additional chances of seeing an acknowledgement before retransmitting their response, the overall load on the network is reduced, leading to a quicker completion time.

Table 1. Time to completion showing several changes to the three algorithms

% loss	Impractical				Successive Linear				Block Adjust			
	None	Repeat	Closer	Both	None	Repeat	Closer	Both	None	Repeat	Closer	Both
00	2228	2228	2166	2303	4746	4746	4701	4711	3054	3054	3020	2964
10	5334	4863	4775	4401	5880	5169	5500	4956	4269	3833	3773	3423
20	6573	5261	5738	4377	7952	5788	6849	5272	5596	4516	4740	3802
30	9002	6676	8026	5197	10336	6433	8365	5935	7719	5447	6050	4419

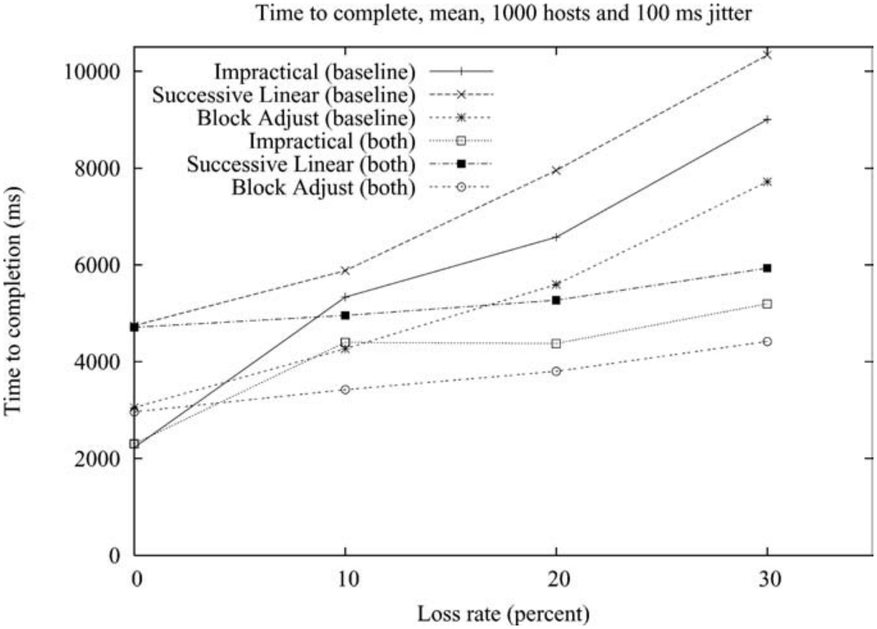


Fig. 5. Finish time showing both enhancements against the unenhanced case

The benefit of repeated acknowledgements is likely to be increased if a greater fraction of the request packet is carrying repeated acknowledgements (i.e., each response is acknowledged a greater number of times). This can be achieved by reducing T_E (since the number of new acknowledgements expected in each request is T_E/I). In fact, lowering T_E may also have a beneficial effect because even though more of the permitted overall load of the protocol on the network is consumed by the requests, the nodes which must retransmit responses discover this sooner; though they cannot do so until the load permits, the overall effect is that the tail is shorter, and completion is quicker. Merely reducing T_E from 200 ms to 100 ms so that requests are sent closer together in time has an effect shown in table 1 under the column "Closer".

Finally, the combined beneficial effect of both changes can be seen in the column header "Both". This is also shown graphically in figure 5.

5 Conclusion

We have introduced new techniques for discovery of nodes which are reliable, scalable, prompt, and robust to loss and jitter. Most importantly, our contribution is not susceptible to a malicious enumerator and cannot be provoked to overload a network. Of these the Block Adjust algorithm is found to be more responsive under realistic conditions.

We also show that using more bandwidth for repeated acknowledgements is overall beneficial to the protocol by reducing unnecessary retransmissions, even though it reduces the bandwidth available to Responses.

References

1. Norm Abramson. The Aloha System - Another Alternative for Computer Communications. In *Fall Joint Computer Conference, AFIPS Conference*, 1970.
2. B. Cain and S. Deering and I. Kouvelas and B. Fenner and A. Thyagarajan. *Internet Group Management Protocol, Version 3.* The Internet Society, October 2002. RFC 3376.
3. Yaron Goland, Ting Cai, Paul Leach, Ye Gu, and Shivaun Albright. *Simple Service Discovery Protocol/1.0.* The UPnP Forum, `http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt`, version 1.0, draft 3 edition, October 1999.
4. H. Schulzrinne and S. Casner and R. Frederick and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications.* The Internet Society, January 1996. RFC 1889, see particularly section 6.2.
5. Mark Handley. Session Directories and Scalable Internet Multicast Address Allocation. In *SIGCOMM 1998*, volume 28(4) of *Computer Communications Review*, pages 105–116, October 1998. See especially section 3.1.
6. Mor Harchol-Balter, Tom Leighton, and Daniel Lewin. Resource discovery in distributed networks. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 229–237. ACM Press, 1999.
7. Roger Kermode and David Thaler. Support for Reliable Sessions with a Large Number of Members. In *Networked Group Communication*, November 1999. First International COST264 Workshop.
8. Microsoft Corporation. *Windows 2000 Server TCP/IP Core Networking Guide*, Appendix I - Windows 2000 Browser Service. Microsoft Press, May 2002. ISBN 0735617988, Also at `http://www.microsoft.com/resources/documentation/windows/2000/server/reskit/en-us/tcpip/part4/tcpappi.mspx`.
9. Puneet Sharma and Deborah Estrin and Sally Floyd and Van Jacobson. Scalable Timers for Soft State Protocols. In *Proceedings of the INFOCOM sixteenth annual joint conference of the IEEE Computer and Communications Societies*, page 222. IEEE Computer Society, 1997.

gTrace: Simple Mechanisms for Monitoring of Multicast Sessions

Gísli Hjálmtýsson, Ólafur Ragnar Helgason, and Björn Brynjúlfsson

Networking Systems and Services Laboratory,
Reykjavik University, Department of Computer Science,
Ofanleiti 2, 103 Reykjavik, Iceland
{gisli, olafurr, bjorninn}@ru.is

Abstract. For multicast to be a viable as a dependable network service, mechanisms for monitoring and managing multicast flows must be developed. Significant prior work exists to use receiver observations to derive properties of multicast groups, including the underlying distribution topology and membership size. However, just as multicast is delegation of replication and distribution from the sender into the network, we contend that similar delegation of monitoring and management into the network is warranted and beneficial. gTrace is a set of mechanisms to monitor multicast distribution trees that operates on routers participating in the multicast distribution to obtain accurate observations throughout the tree in a scalable manner. In this paper we present the protocol and validate its benefits through analysis, simulation and experimentation.

1 Introduction

Multicast offers scalable delivery to large number of receivers by delegating work from senders to the network thereby conserving sender and network resources. However, multicast distribution introduces new and significant challenges. Some of these challenges arise from the inherent properties of multicast that invalidate principal assumptions of unicast Internet protocols. In particular, point-to-point protocols generally assume a closed feedback loop from the sender to the receiver. With multicast this is not the case. In IP multicast models the sender(s) are oblivious to the identity of the receivers and their individual capabilities and observed network performance.

In [1] Sarac et al survey tools for multicast monitoring, classifying them into debugging tools [2,3,4,5,6], management tools [7], and modeling tools [8,9]. Focusing on providing multicast over the MBone, these tools have significant drawbacks as mechanisms to provide consistent level of service quality for multicast sessions. Whereas the debugging tools aim at identifying reachability and loss problems in the MBone, the modeling tools aim at understanding long term behavior of the MBone as an infrastructure, rather than the performance of individual sessions. The management tools again focus on managing MBone connectivity and performance. Some of the problems with these tools include dependence on application layer data, end-to-end monitoring, implosion problems, limited responses, implosion and more. We conclude that all these tools have in common of being infrastructure monitoring tools and do not offer valuable mechanisms for service management.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 264 – 276, 2005.

To offer multicast as a dependable network service it is essential for service providers to be able to monitor service delivery. Service providers exploiting multicast, such as a TV station, or a gaming provider, need to be able to either monitor multicast delivery directly or contract the equivalent of an SLA assuring consistent service quality throughout the multicast tree. Knowing overall loss rate may enable the sender to adapt encoding to reduce bit rate. Identifying bottleneck links may allow congestion adaptation over the bottleneck link, such as local retransmission, transcoding, or selective discard of less important packets. Multicast protocols that implement stochastic feedback depend on knowing (an estimate of) the group size. Knowing group size may help service providers prioritize resource allocation; knowing properties of the distribution topology, including loss rates and bottleneck links helps in identifying and mitigating service delivery problems.

Of particular interest are recent ideas on autonomic networking where a root of a subtree may autonomically activate local retransmission, or may increase forwarding priority in reaction to high losses or long delays respectively. To facilitate, such local reaction, scalable approaches are needed to collect and propagate information such that the relevant information is available throughout the distribution tree.

In this paper we present gTrace, a protocol and mechanisms to monitor multicast distribution trees, combining a request/collect mechanism with local observations at the routers participating in the multicast. gTrace delegates the monitoring and data collection to the routers participating in the multicast to obtain accurate observations throughout the tree while incurring minimal overhead. In contrast to prior work, the protocol focuses on *individual distribution trees* and is applicable to any sparse mode multicast protocol and many application level multicast. Using gTrace all nodes of a given multicast distribution tree efficiently learn key properties of the tree, including topology information, loss rates, the group size, and tree height. While supporting an external observer, gTrace recursively propagates observations throughout the distribution tree. In particular each node in the distribution tree recursively collects (and has at its disposal) information about the downstream subtree. gTrace supports incremental deployment, and remains valuable even if only a fraction of the routers in the multicast distribution tree participate in the protocol.

The primary contribution of this paper is the specification of the gTrace protocol, and its validation using simulation and experimentation. After discussing related work in Section 2, each of the contribution is discussed in successive section. Section 3 describes the protocol, with Section 4 giving examples of usefulness by showing how some key properties can be obtained using gTrace. In Section 5 we evaluate the mechanisms through simulations to show tracking capabilities and to sensitivity analysis of the protocol. In Section 6 we discuss implementation and our experimentation with gTrace in our experimental multicast services for TV distribution and teleconferencing. We then conclude in Section 7.

2 Related Work

As discussed above most multicast monitoring tools have focused on monitoring the MBone infrastructure [2,3,4,5,6,7,8,9]. For example, Mtrace [2] supports tracing the path from a multicast receiver to the source. While potentially valuable, Mtrace is the multicast equivalent of traceroute, does not scale, and is inappropriate for continuous monitoring of multicast. Same applies to MHealth [3], which uses RTCP information to collect the identity of session members before employing mtrace. Another debugging tool, RMPMon [10] relies on RTP monitoring with an SNMP based framework. Mantra[4] collects multicast routing information on the MBONE by querying multicast routers and collecting statistics. Tracetree [11] – a tool to discover the topology of a multicast distribution tree – suffers from implosion as each router in the multicast distribution tree sends a response to the original requestor. While gTrace provides a single set of mechanisms capable of providing all these debug functions (see Section 4), in contrast to the tools above, gTrace is lightweight enough to be used on a continuous basis and provides observations throughout the distribution tree.

MRM [5] generates traffic on designated test groups thus employing active probing to identify faults. HPMM [6] extends MRM to support passive monitoring of actual multicast traffic. HPMM arranges testers into a hierarchy to aggregate responses to faults in the multicast infrastructure to prevent implosion at the observer. gTrace employs similar while more general aggregation function to prevent implosion. Unlike [6] gTrace does obtain information from routers directly, providing more accurate information. Assuming comparable deployment of HPMM and gTrace, gTrace can be viewed as a generalization HPMM, while significantly more lightweight.

Another use of multicast and multicast monitoring is to use end-to-end measurement of multicast traffic as a method to infer internal network characteristics [12] and the logical topology of multicast trees [13]. Probes are sent from the source (root) to the multicast receivers (leaves). The end-to-end loss of probes is used to infer the logical multicast topology and the loss rates of the logical links in the topology. A significant drawback of these methods is that they only compute long term averages and do not adapt well to membership changes. In contrast, gTrace eliminates this guesswork as is obtains observations at routers participating in the multicast. Moreover, gTrace promptly tracks membership and topology changes, and can accurately pinpoint bottlenecks links and fate sharing subtrees.

Sharing some of the same characteristics, is the body of work estimating the size of a multicast group [14,15,16] where members of a multicast group periodically, or on demand, send a probabilistic acknowledgement to an observer. In contrast in gTrace, rapidly tracking the number of multicast participants in *every* subtree of the multicast topology is one of the elementary variables maintained.

3 The gTrace Protocol and Mechanisms

GTrace consists of mechanisms and a protocol to perform three main functions: 1) local maintenance and information collection at the multicast routers, 2) a distribution mechanism to propagate requests and information from a collection point to the multicast nodes of the topology, and 3) information gather to propagate information from leaves in the topology towards a collection point applying a summation function at each intermediate node.

To simplify the discussion, in this section we will assume that the collection point is the root of the multicast distribution tree being monitored. Each node in the tree apart from the root, has exactly one upstream neighbor and zero or more downstream neighbors. The gTrace state is identified by the root of the multicast tree and a group identifier. In single source multicast [17,18] this is the channel identifier. In PIM-SM and other shared tree approaches, the root of the multicast could be the rendezvous point, or the root of any of the source specific trees. The local state at each node mirrors that of multicast forwarding, having a downstream state for each downstream neighbor. However this state is control plane state and stored in regular memory and hence not size critical.

An external management system may employ gTrace. This is indeed how we see typical use by service providers, and how we have employed gTrace for service monitoring and management in our experimental services. However, this can be achieved by periodic report requests or by establishing an association between the external observer and a collection point within the distribution tree, typically the root of the multicast distribution tree.

Although in this paper we focus on gTrace for a single source distribution trees, we have experimented with extension to gTrace for more general topologies and overlays [19].

3.1 Local Information Collected at Intermediate Nodes

The local information collection mechanism interfaces with the multicast topology management module to collect local multicast properties of interest. The topology management module is queried for active multicast flows and returns a list of active flows, each identified by the pair <*S,C*>.

For each multicast channel being monitored, gTrace maintains information about upstream neighbor, and set of next-hop downstream neighbors. For each downstream neighbor, the local state maintained by gTrace consists of the weight (w) i.e., total number of nodes, number of leaves (l), the height (h) of the subtree rooted by that neighbor and the true height (i) measured in physical hops based on TTL. Space complexity at each node is therefore four integer variables times the local fan-out of each monitored channel.

In addition gTrace can obtain from the multicast daemon the number of bytes (b) and packets (p) received on the flow <S,C>. We have defined an abstract interface through which the gTrace daemon queries the multicast topology module to export its state to the local information collection mechanism. This interface must be adapted to the particular multicast protocol in use.

3.2 gTrace Protocol Messages

We have defined three types of gTrace messages, a query, a report, and an update. A query message is used to subscribe to periodic updates of the query result. The report message is used to propagate information down the tree from the originator. The update is used by the gather mechanism as a response to a query.

The format of the gTrace message consists of a message type, type header, a data descriptor, and data values. The message type is one of query, report, or update, and is encoded as a single character. The type headers are described in the respective subsections below. The message data descriptor is a null terminated string of characters, each character value corresponding to a particular variable of interest. The six basic local state variables – *w*, *l*, *h*, *i*, *b* and *p* – are encoded with the corresponding character. In addition the messages may have two fields for IP address and TTL value, encoded by *a* and *t* respectively. The values follow the string (at the next 32 bit boundary in our implementation) with the variables appearing in the same order as they appear within the string. All variables are encoded as 32 bit values, except the byte and packet counts which use 64 bits.

3.3 The Distribution Mechanism

The distribution mechanism is used to distribute queries and information from the collection point to the nodes of the topology. The originating node creates a distribution message and sends it on the multicast group. Distribution messages are sent with the router-alert (hop-by-hop) option down the multicast tree. Distribution messages are sent unreliably, but repeated three times on each link to overcome losses.[1]

Query Subscriptions: The query message is used to subscribe to the continually updated computation of results specified by the message descriptor. The query message type header specifies an update period. The update period is given in milliseconds between updates. A null value in the update period cancels prior subscriptions if one exists. In addition a query always has the *a* and *t* variables specified. A gTrace node that receives a query sends the state values specified in the query message to the upstream gTrace node given as the value of the address field. Before forwarding the query message downstream, the node stamps its own address and the current TTL in the *a* and *t* fields of the message. Unless the update period is zero it then continues to send updates periodically, and stores the query locally. When a new branch joins the topology the query is forwarded on the new branch, extending the subscription to the newly joined downstream nodes.

Explicit Reporting: The report message is used to propagate information down the tree, or report to an outside observer. The type header contains a report-to field containing an IP address, to which reports are to be sent, and a replace/trace flag, encoded as *r* or *x* respectively. If a trace is requested (i.e., r flag set), the processing of the report message is similar to the processing of a record route IP option, in that each node in the multicast distribution tree appends its values for the variables specified in the option header field before forwarding it on each of the output ports of the channel. Each subsequent hop in the path processes the message by appending one element to the array, containing the selected variables. When the report reaches a leaf (or the last hop gTrace router), a single result is sent to the report-to host.

If instead the replace flag is set, each node in the path replaces the values in the message with its own local values before forwarding the message downstream and to the report-to host. If no trace flags are set the original values are propagated to all

[1] If the three messages are sent sufficiently far apart, losses are independent. Even at relatively high loss rates, e.g. 8%, the probability of loosing all three is insignificant. As messages fit in a single packet, overhead is negligible.

nodes below the originator in the topology. For example, this is how the root may announce the group size to all receivers.

3.4 The Gather Mechanism – Processing Subscriptions

The gather mechanism propagates information from the leaves of the topology towards the root by periodically sending updates upstream. At each node the gather involves two functions: a) Receiving and processing update messages from downstream, and b) preparing and sending of an update result upstream. The two functions are performed asynchronously.

The values in a gather message received from downstream on a given channel is simply stored as part of the channel gather state, and override any previous updates from the same downstream node. The message format of the update is the same as of the originating query subscription.

Periodically, for each multicast flow, each router computes a new local state and sends an update to its upstream router. The new local state is computed from local observations and from the state updates from downstream neighbors of the multicast topology. The number of bytes and packets, b, and p, received on the multicast from source S are observed locally. The remaining values, w, l, h, and i are computed by applying an appropriate summation function respectively as

$$w = 1 + \sum_{j \in out(v)} w_j \qquad l = \sum_{j \in out(v)} l_j$$

$$h = 1 + \max_{j \in out(v)} \{h_j\} \quad i = \max_{j \in out(v)} \{i_j + \Delta(TTL)\}$$

where *out*(v) denotes the set of output ports for the multicast, and Δ(TTL) is the difference in TTL of the IP header of the message and the t value in the message.

For each active flow, gTrace includes a timer field that denotes the time when the next local update is to be sent upstream, and for the output ports denotes the time when a state update was last received. If the state update is managed as part of refreshing the multicast forwarding state these timers are superfluous, and omitted.

To economize on message processing, the state update may be sent as part of the message(s) for refreshing the multicast forwarding state. Alternatively, the state updates of multiple multicast groups may be combined and reported in a single message. For a particular flow the updates are sent asynchronously and independently. The update messages are sent upstream on the incoming port of the multicast flow.

The gather process starts at the leaves or at last-hop routers. If the end-systems do not participate in gTrace the last-hop router sets $h = i = 1$ and may attempt to estimate the appropriate values to w and l, depending on the information that it has available from the local collection mechanism (which may be protocol dependent). Without superior information, the router sets l to its local outdegree (for the group), and w to $l + 1$. To gracefully leave a session when a node leaves the multicast group the node sends a last gTrace message with $w = l = h = i = 0$.

An important tradeoff in the realization of the gather mechanism is the length of the update period. A short update period gives better state estimates at the increased overhead cost of bandwidth and processing at the nodes. The update period can be adjusted by the collection point by resending the query with a new update period. In

Section 5 we evaluate the gTrace gather mechanism through simulation to evaluate the effect of the update period on the estimation lag and error.

All gTrace messages are sent unreliably. However since an upstream node simply maintains the last received update as its current best estimate (as long as the multicast branch is active), under reasonable loss rates, losses merely delay the propagation of information updates. Of course this contributes to errors in the estimates of the various values, as discussed in the next two sections.

3.5 Incremental Deployment

Initially during the deployment of the gTrace mechanisms, one cannot assume that all multicast routers participate in the protocol. A router that does not implement the gTrace protocol will simply forward report messages without processing. This is not a major issue as the distribution messages are forwarded on the multicast group, and thus all routers and receivers will receive the message. Updates are sent point-to-point to the address of the closest upstream gTrace router learned from the query message. In fact, by comparing the TTL value of the message to that of the current TTL in the IP header of the packet containing the query message the downstream gTrace nodes can determine how many non-cooperating routers there are between it and the next upstream gTrace enabled router.

4 Computing Multicast Attributes

In this section we describe how to use gTrace to compute a number of interesting properties of multicast. We consider two types of applications, group centric, and network centric. The former is used by application level protocols or a group manager to adapt and optimize the behavior of the application level protocols. The latter is used by network and service management systems to monitor and manage resources allocated to the multicast groups, and apply network policies for resource sharing. The application sets the gTrace options depending on the needs of the group. Note that although the data collection is coordinated, it is still an estimate due to staleness caused by protocol delays and the group dynamics.

Estimating Group Size: The leaf count *l*, at the collection point (root of the distribution tree) is the current estimate of the group size. More generally, the leaf count at any intermediate node is the down-stream group size. For some applications (e.g. applications using RTP) it is important for the leafs to know the group size. This information can be propagated to the receivers by sending a gTrace report from the multicast root. The source records its own information, source address, weight, height and number of bytes sent. Without any flags set the intermediate routers simply forward the message on the multicast tree. On receipt, receivers learn the current estimated group size.

Identifying and Locating Bottleneck Links: Since number of packets received is collected and exchanged, gTrace can be used to identify a bottleneck link in a multicast distribution tree. Each node periodically transmits upstream the number of packets it receives on a flow. By comparing the local receive rate to the receive rate on a given downstream branch (from the gather update), a node can determine if the losses on the downstream link are "excessive." This information could be used to activate local retransmissions on that particular link or employ other forms of local congestion adaptation [20].

A more thorough analysis of losses and in fact topology can be obtained by using a report message with the record flag to propagate and collect the addresses and received packets of every node at a service management host (specified in the report-to field) for analysis.

Determining the Number of Receivers Sharing a Bottleneck Link: Using the above and including the weight, and/or leafs in the subscription the nodes above the bottleneck knows the number of node and/or leafs below that link. A similarly augmented report message, i.e. requesting weight and leafs, can be employed to deliver that information to an external observer.

Topology Discovery: We can use gTrace to discover the topology of the multicast in a number of ways. Sending a report with the *trace* flag set and address field selected, propagates the traceroute from the root to each node in the tree. By specifying a *report-to* this information will be sent from each leaf to an external observer. To appreciate the underlying topology, including the TTL value as well is valuable. Using the same with the *replace* flag implements essentially the same semantics as [11].

5 Evaluation

Although the protocol is based on direct measurements the aggregate values seen at the collection point are the result of a distributed computation by the gather mechanism with information exchanged over an unreliable channel. Consequently the values at the root are (random) estimates of the true values. There are three sources of errors: staleness – since the values are observed at the root some time after the observations were collected; inconsistent observations – since the aggregate is computed from observations made asynchronously; and losses of update messages. In this section we use simulations to examine the dynamic behavior of gTrace, in particular its ability to track dynamic changes in topologies. For this purpose we have built a simulation model of gTrace mechanisms using the ns2 network simulator. Analysis of the steady state distributions is found in [21].

5.1 The Basic Model

In each simulation run we construct a multicast topology and a finite set of potential receivers. Receivers are either busy, or idle, becoming busy by joining a multicast group and idle when leaving the group. We use a two phase model with exponential holding times to model client activity.

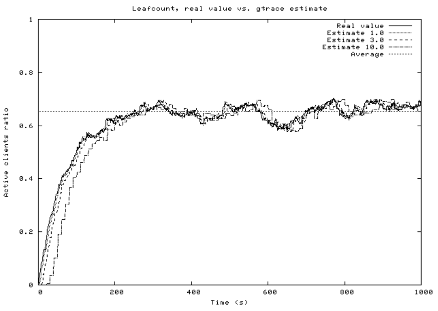


Fig. 1. Dynamic tracking of gTrace using update period of 1, 3 and 10 seconds, over a full binary tree of height 8

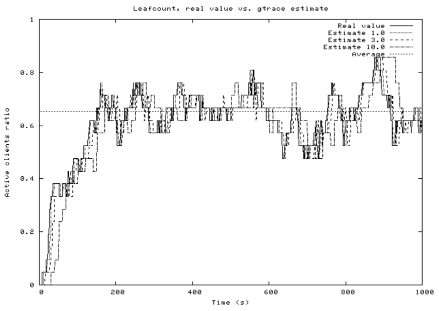


Fig. 2. Dynamic tracking of gTrace over a random tree of height 10 with 21 potential receivers

We have used both regular topologies (binary- and k-ary trees) and random topologies. The random distribution trees are constructed in a recursive manner as follows: Each node has some probability αleaf of being a leaf. If not a leaf, the fan-out of the node is determined as a random variable. We use a geometric distribution with mean 1.8 to generate sparse trees, and a normal distribution with mean 2.5 to generate more dense trees. To limit the size of the tree we limit the maximum height of the tree.

In our simulations we assume an end-to-end loss rate of approximately 4% from the deepest leaf to root. Each link has the same loss rate and losses are independent and identically distributed. Ignoring the processing overhead at nodes we use the same propagation delay for all links, 20 ms. While this is rather long, it is still an order of magnitude smaller than the smallest update period we consider.

5.2 The Ability to Track Dynamic Changes in Topology

To evaluate how gTrace handles dynamic topology we use the model described above with expected idle time of 100 sec, and expected active time of 200 sec. Each receiver joins and leaves according to this process repeatedly throughout the entire simulation.

Figure 1 shows a simulation run for a full binary tree of height 8 (thus having 256 leafs). The figure plots four curves, the true number of active receivers and the corresponding gtrace estimate (leafcount) at the root of the topology for three different update periods: 1, 3 and 10 seconds. To facilitate comparison with other figures, the y-axis is normalized to show the fraction of potential receivers active. As is evident form the figure, gTrace tracks the actual value excellently even with the 10 second update periods.

Figure 2 shows the same for a random tree of maximum height 10 having 72 nodes, thereof 21 leafs, created using αleaf = 0.5 and a geometrically distributed fan-out. The significantly fewer leaves result in substantially less activity, accounting for the steps in the curves. Again, we see that gTrace accurately tracks the actual membership.

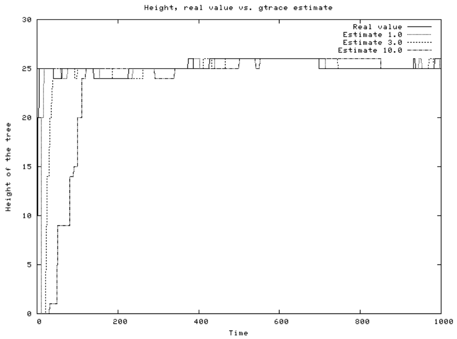


Fig. 3. Dynamic tracking of height in gTrace over a random tree of height 26 with 150 potential receivers

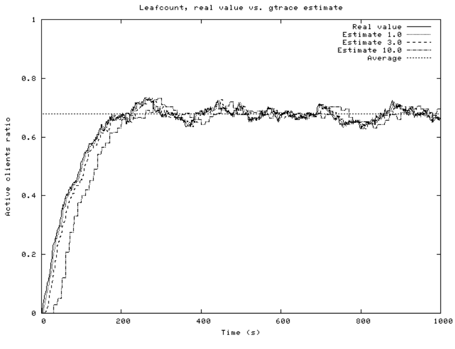


Fig. 4. Dynamic tracking of over a random tree of height 8 with 279 potential receivers

gTrace ability to track the height of a tree is evident on Figure 3 which shows a simulation run for a random tree of having 447 nodes, thereof 150 leafs, the furthest leaf having height 26. The tree was created using αleaf = 0.3 and a geometrically distributed fan-out. The client behavior is the as before except using expected idle time of 200 sec, and expected active time of 100 sec to get more dynamic in the height. While the significant height results in appreciable lag for the 10 second update time, the figure shows that gTrace accurately tracks the height.

5.3 Insensitivity to the Underlying Topology

Figure 4 shows again the dynamic tracking of gTrace for a denser tree topology. The tree was created with $\alpha_{leaf} = 0.5$ and a normally distributed fan-out with expected value of 3, and maximum height set to 8, yielding a tree with comparably many receivers as the full binary tree in Figure 1, namely having 279 leafs out of a total of 690 nodes. Comparing Figure 1 and Figure 4 we see how gTrace is indifferent to the underlying topology as the figures are almost identical.

5.4 Effect of Tree Height on Estimates

Figure 5 and 6 show the effect of height on the lag in information updates, depicting two scenarios with equal number of clients but with different height. For Fig. 5 the tree topology is a full binary tree with 512 receivers and thus height of 9. Fig. 6 shows a full 8-ary tree topology with 512 clients and a height of 3. For the binary tree and update period of 10 seconds the lag is clearly visible while the difference for the 8-ary tree is negligible.

6 Implementation and Experimentaion

We have prototyped and experimented with gTrace for monitoring and managing multicast services, such as TV distribution and teleconferencing that we are

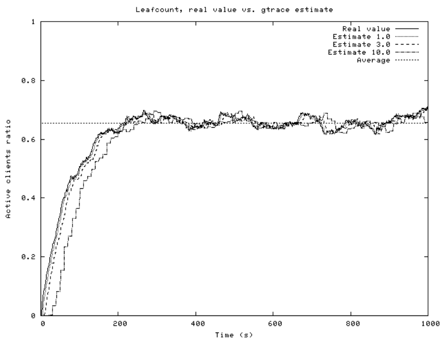


Fig. 5. Dynamic tracking of gTrace over a full binary tree of height 9 with 512 potential receivers

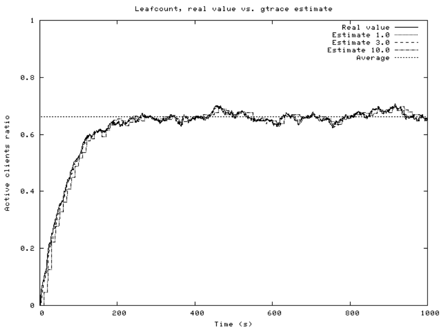


Fig. 6. Dynamic tracking of gTrace over a full 8-ary tree of height 3 with 512 potential receivers

with over the Internet. Our multicast services are offered over our own novel single source multicast protocol, SLIM [18]. Initiated from our service management center we use the gTrace mechanisms to collect statistics about use and to monitor changes in membership and topology.

Local Information Collection at Intermediate Nodes: A gTrace local collection object communicates with the topology management daemon of SLIM to collect the local state information. We have implemented the abstract interface using shared memory minimizing IPC overheads.

The Distribution and Gather Mechanisms: On receiving a query the daemon sets up the appropriate measurement state and an update timer value as specified by the message descriptor. When a new branch joins the multicast tree the daemon forwards the last query out on the branch. When a report message arrives the daemon serves the report by appending appropriate values, forwards the message and sends a copy to a report-to host if applicable. A gather message from a downstream node triggers the daemon to update its estimates for that particular sub-tree. On update timer expiration for a particular group the local state is collected from the SLIM daemon as well as the current estimates from downstream nodes and are then sent to the upstream router.

Experimentation with Multicast Services: We have been offering multiple multicast based services based on our SLIM protocol, including TV distribution, radio and conferencing on experimental basis for over two years. In particular we offer popular TV channels not available in remote areas in Iceland, at times received by hundreds of concurrent users outside of our campus. gTrace is proving lightweight and efficient in supporting the monitoring of multicast in this setting.

7 Conclusion

In this paper we have introduced gTrace, a control plane mechanism for measuring and monitoring multicast topologies. The gTrace control plane protocol consists of a distribution and gather mechanism, and three types of messages. Collectively the protocol allows for observations from *inside* of the multicast topology to be combined in a scalable manner to compute a flora of interesting attributes of multicast distribution trees. Our simulations show that gTrace is able to track key attributes accurately and is relatively insensitive to topology and dynamic member changes. This we have indeed verified in our own use of the protocol in our experimental multicast service for TV distribution and teleconferencing.

References

1. K. Sarac and K. Almeroth, "Supporting Multicast Deployment Efforts: A Survey of Tools for Multicast Monitoring", Journal of High Speed Networking - Special Issue on Management of Multimedia Networking, vol. 9, num. 3/4, pp. 191-211, March 2001.
2. Bill Fenner and Steve Casner, “A traceroute facility for IP multicast”. Internet Draft, work in progress, Internet Engineering Task Force, July 2000.
3. D. Makofske and K. Almeroth, "MHealth: A Real-Time Multicast Tree Visualization and Monitoring Tool", Network and Operating System Support for Digital Audio and Video (NOSSDAV '99) , Basking Ridge New Jersey, USA, June 1999.
4. P. Rajvaidya and K. Almeroth, “A Router-Based Technique for Monitoring the Next-Generation of Internet Multicast Protocols”, International Conference on Parallel Processing (ICPP), Valencia, Spain, September 2001.
5. K. Almeroth and L. Wei, “Multicast Reachability Monitor”, IETF Internet Draft, work in progress, July 2000, draft-ietf-mboned-mrm-01.txt
6. J. and B. N. Levine, “A Hierarchical Multicast Monitoring Scheme” in International Workshop on Networked Group Communication (NGC), Palo Alto, California, November 2000.
7. D. Thaler, @Globally distributed troubleshooting (GDT): Protocol specification,” IETF draft, draft-thaler-gdt-*.txt, January 1997.
8. K. Almeroth and M. Ammar, “Multicast group behavior in the Internet’s multicast backbone (Mbone),” in IEEE Communications, vol. 35, pp. 224-229, June 1997.
9. M. Yajnik, J. Kurose, and D. Towsley, “Packet loss correlation in the MBone multicast network,” in IEEE Global Internet Conference, London, UK, Nov. 1996
10. J. Chesterfield, L. Breslau, W. Fenner, “Remote Multicast Monitoring Using the RTP MIB”, in the proceedings of MMNS '02, Santa Barbara, 2002.
11. K. Sarac and K. Almeroth, "Tracetree: A Scalable Mechanism to Discover Multicast Tree Topologies in the Network", accepted for publication in IEEE/ACM Transactions on Networking.
12. N.G. Duffield, F. Lo Presti, “Multicast Inference of Packet Delay Variance at Interior Network Links”, in Proc. IEEE Infocom 2000, Tel Aviv, Israel, March 26-30, 2000
13. N.G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley, “Multicast Topology Inference from Measured End-to-End Loss”, IEEE Trans. on Information Theory, vol. 48, pp. 26–45, 2002.
14. J.-C. Bolot, T. Turletti, and I. Wakeman, “Scalable feedback control for multicast video distribution in the Internet”. In Proc. of ACM SIGCOMM’94, London, UK, pages 58–67, September 1994.
15. S. Alouf, E. Altman, P. Nain, "Optimal on-line estimation of the size of a dynamic multicast group". In Proc. IEEE Infocom 2002, New York, USA, June 2002.

16. T. Friedman, D. Towsley, "Multicast Session Membership Size Estimation". In Proc. IEEE Infocom'99, New York, USA, March 1999.
17. Holbrook H., and D.R. Cheriton, IP multicast channels: EXPRESS support for large-scale single-source applications, in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 1999.
18. G. Hjálmtýsson, B. Brynjúlfsson and Ó. R. Helgason, "Self-configuring Lightweight Internet Multicast", in proceedings of IEEE SMC 2004, Hague, Netherlands, October 2004.
19. Gísli Hjálmtýsson, Ólafur Ragnar Helgason and Björn Brynjúlfsson, "Simple Active Mechanisms for Measuring and Monitoring Service Level Topologies", in proceedings of IWAN 2004, Lawrence Kansas, USA, October 2004.
20. Gísli Hjálmtýsson and Samrat Bhattacharjee, “Control on Demand – An Efficient Approach to Router Programmability,” in IEEE JSAC, Vol. 17, No. 9, September 1999, pp. 1549-1562.
21. Gísli Hjálmtýsson, Ólafur Ragnar Helgason and Björn Brynjúlfsson, "Analysis of gTrace", Reykjavik University Technical Report, NL200404, October 2004.

Toward Feasibility and Scalability of Session Initiation and Dynamic QoS Provisioning in Policy-Enabled Networks

Kamel Haddadou[1], Yacine M. Ghamri-Doudane[2], Marc Girod-Genet[3], Ahmed Meddahi[4], Laurent Bernard[3], Gilles Vanwormhoudt[4], Hossam Afifi[3], and Nazim Agoulmine[1]

[1] LIP6 Laboratory, Pierre & Marie Curie University, 8, rue du Capitaine Scott,
75015 Paris, France
{Kamel.Haddadou, Nazim.Agoulmine}@lip6.fr
[2] Computer Engineering Institute (IIE-CNAM), 18 Allée Jean Rostand,
91025 Evry Cedex, France
Ghamri@iie.cnam.fr
[3] GET/INT, Network and Telecommunication Service Department, 9 rue Charles Fourier,
91011 Evry Cedex, France
{Marc.Girod_Genet, Laurent.Bernard, Hossam.afifi}@int-evry.fr
[4] ENIC Telecom LILLE 1, rue Guglielmo MARCONI,
59658 Villeneuve d'Ascq Cedex, France
{Meddahi, Vanwormhoudt}@enic.fr

Abstract. In this paper we implement and evaluate a new solution for the multimedia session setup with resource allocation in policy-enabled networks that we developed in [1]. Our proposal is based on the use of Session Initiation Protocol (SIP) in the framework of Policy-Based Management (PBM)[1]. We specifically evaluate the feasibility and the scalability of such solution in a real environment through experimentation on a test-bed. The latter integrates all the components from session initiation to QoS policy enforcement into network elements. Our results demonstrate both of the relevance and the efficiency of such solution.

Keywords: SIP, PBM, Dynamic QoS Provisioning & Signaling.

1 Introduction

A major challenge in emerging multi-service, QoS-capable telecommunication networks is the deployment of high-quality multimedia applications. Both of network operators and end users are willing to offer and use multimedia communications with a large range of QoS-guarantees. To achieve this aim, an efficient control and management of network resources are submitted to be the key issues in the telecommunica-

[1] This work is undertaken within the MMQoS Project founded under the RNRT/French Ministry of Industry program.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 277 – 288, 2005.

tions world. So, we argue that a combination of end-to-end signaling and Policy-Based Management (PBM) [2] is required to enable proper multimedia sessions.

PBM aims to facilitate the management activity as it allows network administrators to define high-level objectives of network management schemes based on a set of policies. This latter is a set of pre-defined rules controlling network resources. Rules, established by the network administrator, include actions to be triggered when a set of conditions is fulfilled. PBM approach allows in its turn the translation of these high-level rules to a set of low-level device-compliant configuration commands.

As for signaling, SIP [3] is gaining increasing momentum as a protocol that enables set up of multimedia sessions. It has been adopted by both the Internet Engineering Task Force (IETF) for IP-based networks, as well as the 3rd Generation Partnership Project (3GPP) for next generation mobile networks.

Increasing the session establishment flexibility, while dynamically controlling the access to the resources, makes it thus easier to guarantee QoS in multi-service networks. In our work referred in [1], we proposed a novel approach that integrates session establishment with dynamic QoS control. We specifically transfer parts of the network management and control mechanisms to the user's terminal.

In the current work, we analyze the feasibility and the scalability of this approach taking into account session-oriented QoS parameters, mainly the delay of session establishment, as required in ITU-T recommendations [4].

We first integrate session establishment with dynamic resource allocation. In other words, we relate how to link session setup using SIP to QoS management using PBM. We then transfer numerous signaling and management functions to the user's terminal.

The rest of this paper is organized as follows: Section 2 introduces SIP and PBM. Section 3 presents our solution and some main related features. In Section 4, we describe the test-bed on which we implemented our solution. The empirical results targeting several feasibility and scalability testing scenarios are analyzed in Section 5. Finally, Section 6 concludes the paper and presents some future works.

2 Background

As our solution is based on SIP and PBM integration, these technologies are described briefly in the following sub-sections:

2.1 The Session Initiation Protocol (SIP)

As it was standardized by the IETF, SIP [3] can be considered as an application-layered signaling protocol. Its main role is to set-up sessions or associations between two (or occasionally more) Internet users or systems. The sessions that are initiated with SIP can be used to exchange various types of media. Specifically, SIP sessions are commonly used for handling voice media over packet networks. SIP is a client/server-oriented protocol with two types of messages: requests and responses. Messages are encoded in textual format using a structure similar to the HyperText Transfer Protocol (HTTP). It defines several messages to request action from the server. One of these messages is the INVITE message that is sent to invite another participant to a session, the other is the BYE message for closing a session, and finally the ACK message to confirm session establishment. The response contains a status code to indicate the success or the failure of the request (e.g. 200 OK for establishment success). Message's body can also contain media or session description. Hence, when establishing a session, SIP exchanges also media attributes in order to share a common set of capabilities.

In SIP based network architecture, the previous messages are exchanged between four main types of entities playing different roles: User Agent, Proxy Server, Redirect Server and Registrar server. User Agents (UA) are SIP endpoints that send (caller) or receive (callee) signaling messages. A UA is divided in two components, the first acts as Client (UAC) and initiates the sessions; the second acts as a server (UAS) which is responsible for replying the session initiators. A UA communicates with another one directly or via intermediate proxy and redirect servers.

Proxy servers are application-layer routers that forward SIP requests and responses. Redirect servers receive requests and then return the location of the targeted SIP UA or location of the server where this user might be found. Registrars keep track of participant information (correspondence between SIP and IP address, access rights…). Proxy or redirect servers use registrars to determine routing or participant policies.

2.2 PBM Architecture and COPS Protocol

The end to end negotiation process, offered by SIP, only ensures capability exchanges. To allow Quality of Service (QoS) provisioning in an IP backbone, capability exchange is not sufficient and resource reservation mechanisms have also to be considered. In the current research initiatives, that are undertaken in order to simplify network resource management, the PBM [2] approach is the one which gains more interests due to the important number of advantages that it offers.

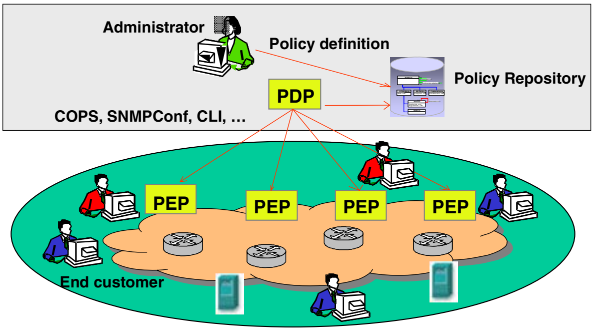


Fig. 1. PBM components

The IETF Resource Allocation Protocol (RAP) Working Group has specified a scalable and secure framework for policy definition and administration [5]. This framework introduces a set of components to enable policy rules definition, saving and enforcing: the Policy Enforcement Point (PEP), the Policy Decision Point (PDP) and the Policy Repository (Figure 1). PEP components are policy decision enforcers located in network and system equipments. The PDP is the component responsible for high-level decision-making process. This process consists of retrieving and interpreting policies, and implementing the decision in the network through the set of PEPs. The policy repository contains policy rules that are used by the PDP. To describe policies and network information, the IETF has adopted the Common Information Model (CIM) [6], which is a neutral scheme implementation describing overall management information.

In order to exchange management information and/or decisions, the PDP interacts with each PEP using one of the several protocols specified or extended for this purpose. Among them, the Common Open Policy Service (COPS) protocol [5] is the one which was designed specifically by the IETF to realize this interaction.

2.2.1 COPS Protocol

COPS is a client/server protocol allowing the exchange of policy information between a PEP and its corresponding PDP. This exchange is realized through three main messages: the request (REQ), decision (DEC) and report (RPT) messages. Hence, after a connection establishment between the PEP and its serving PDP, the PEP transmits requests for decisions to the PDP using the REQ message. In response to a REQ, a decision message (DEC) is sent by the PDP. Then, the PEP reports the outcomes to the PDP via the RPT message.

Initially, the COPS protocol was designed mainly for resource allocation in an Internet backbone. In order to make such a reservation, two models within the COPS protocol were proposed: the Outsourcing and Provisioning models. In the former policy-requests are triggered by particular events when in the latter policies are installed in the PEP before the PEP decides how to treat the event. For this second model, a specific COPS extension, called COPS PR [7], have been designed.

In order, to facilitate the resource allocation process and render the PBM approach more dynamic, the authors in [8] proposed using the COPS protocol to unify both QoS signaling and resource allocation. The idea behind this is that each end-system will encompass a specific PEP that have to initiate resource reservation requests. Requests are handled by the PDP which in its turn takes decisions according to current resource usage and customer's Service Level Agreement (SLA).

In our work, both QoS signaling by end-system and dynamic resource provisioning in edge routers are considered. In this case, each time a new reservation request is accepted by the PDP, the latter both provisions the concerned edge router accordingly (using COPS PR) and informs the end-system on its positive decision. For end-system QoS signaling, our proposed COPS extension is called COPS usage for QoS Parameter Signaling (QPS). More details on COPS QPS operations can be found on [9].

3.1 Functional Architecture

As shown in Figure 2, our solution is based on the integration of SIP proxy and QPS PEP. This environment allows relocating networks entities inside the terminal and then can be seen as a network extension. In [1] we demonstrate the advantage of this integration by comparing it with other proposals based on the combination of session initiation and QoS provisioning. Actually, one can note that the integration of the SIP proxy and the QPS PEP allows a complete control of session-setup by the network. Indeed, the SIP Proxy participates to every step of SIP exchange. Then, it knows all the information concerning the session. Furthermore, its implementation near to the QPS PEP automates the interactions between application's needs and network-resource allocation. As part of network management functionalities (QPS PEP and SIP Proxy) are deported into the user's terminal, these functionalities have to be ran within a secured environment (execution environment secured by smart card [1][10], for example). How to secure such an environment is not the target of our paper and will not be described.

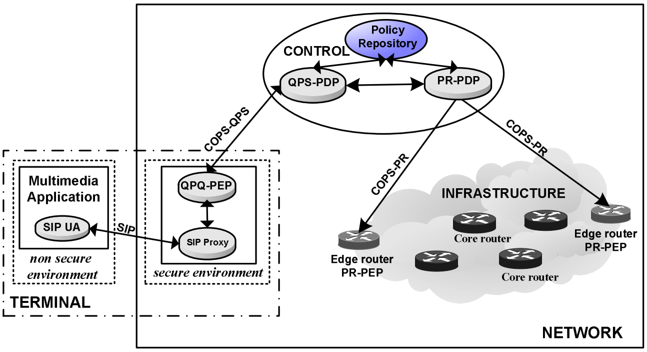


Fig. 2. Functional Architecture

3.2 Session Establishment Sequence Diagram

The complete process of QoS-enabled session establishment process is detailed in this section (Figure 3). At the beginning, the UAC initiates a session by sending an INVITE message to its correspondent UAS. This message is intercepted by the client's SIP proxy. After having recovered the media information concerning the SIP session initiator, the SIP proxy forwards the message to the UAS. When the UAS replies with a 200 OK, the SIP proxy recovers the media characteristics concerning UAS. At this stage, it has all media information concerning the media session. The SIP proxy is then responsible of translating this information into QoS parameters (bandwidth, delay, jitter and packet loss) and sending them to the QPS PEP. Using these QoS parameters, the QPS PEP sends a QoS request to the PDP. First of all the PDP consults the user's SLA and the resource availability before generating the appropriate decisions towards the concerned edge-routers and the caller's terminal.

Note that, each time a COPS QPS REQ message arrives to the PDP, the policy repository is accessed three times: the first is for the retrieval of the client's SLA, the second is to retrieve admission control policies and the last access allows retrieving topology information. This latter is realized in order to identify the edge-routers that will be crossed by media flows. The number of accesses realized after the reception of a COPS QPS REQ message depends on the structure of the policy repository that is a realization of the CIM model [6]. These accesses are optimized as we have three information to retrieve and we access these information directly. Indeed, the policy repository is realized as a Lightweight Directory Access Protocol (LDAP) [11] server. When using LDAP we have the ability to access information directly using their distinguished names (dn). Hence, in our architecture, we use, as distinguished names (dn), specific information related to terminal location (SubNet Address, Wireless Cell Identification or BSSID, …), that are sent as objects of the QPS REQ message. The use of dn accelerates substantially the delay of each information access [12].

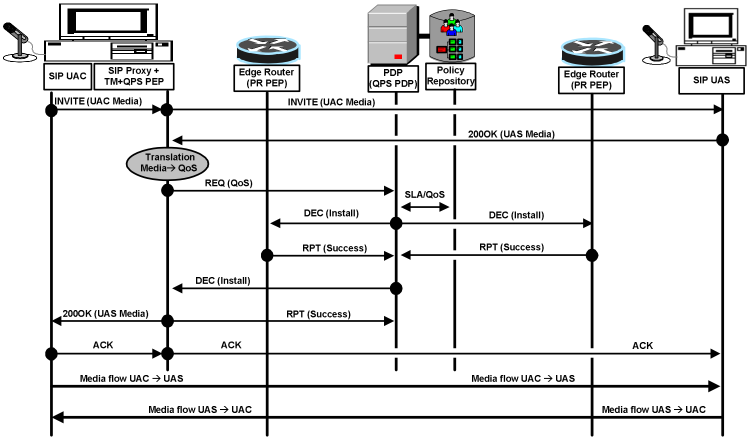


Fig. 3. Session establishment phase with QoS setup

The decision that is sent by the PDP to the crossed edge-routers resides on classifying and marking media-flow packets with the appropriate QoS class tag [13]. Once the successful enforcement of the decision is reported to the PDP, the latter sends the appropriate decision to the requesting QPS PEP. In the case of acceptance, the SIP 200 OK message is directed to UAC which terminates the session setup. The multimedia application can starts flow transmissions tacking advantage of the QoS-level assigned to it within the network.

4 Test-Bed

As we intend to verify the feasibility and scalability of our original solution, an integrated test-bed, containing all SIP and all PBM entities, is carried out. Unlike other experiments found in the literature [1], the COPS operations are not considered alone but all PBM architecture components are included to our test-bed. Indeed, the interactions between PDP and the policy repository, policy interpretation by the PDP, and real policy enforcement are implemented. The details of our test-bed building blocks are given bellow.

4.1 SIP Building Block

All SIP entities, listed above, have been implemented using the Java-based specifications called Jain-SIP [14].

- SIP User Agent Client: this component is the one that generates SIP requests and records call set-up delays (from the "INVITE" message up to the "200 OK"),
- SIP Proxy: in addition to its classical operations defined in SIP [2], it is in charge of the interconnection with the Translation Module (TM) (cf. Section 4.3),
- SIP User Agent Server: this component generates automatically the responses to UAC requests.

4.2 PBM and QoS Building Block

The set of PEPs, the PDP, and the COPS protocol have been implemented using Java. As denoted previously (cf. section 3.2), the policy repository is realized as a LDAP schema, where in the management information are modeled using CIM. The distinguished names (dn), in the LDAP schema, are chosen to correspond to objects transported by incoming COPS messages. This allows rapid and direct access to the management information that is needed to handle a new media flow.

In addition to PBM components, specific software, the Traffic Designer [15], is used for the enforcement of QoS decisions (traffic classification and packet marking) within edge routers.

4.3 Translation Module (TM) Building Block

- The TM building block is mainly responsible of converting the media information into QoS requirements. It interconnects SIP building block to PBM building block. More specifically, this building block allows converting and relaying information between the SIP proxy and the QPS PEP. In our test-bed, this is realized throughout a simple conversion table built thanks to some tests of the behavior of the used codecs. A more accurate mapping outwards the scope of this paper and will be a subject of a future work.

5 Tests and Measures

Our aim is to ensure that the delays of the session establishment due to our proposition remain always under the recommended ITU-T delay limit of 6s [4]. This limit is given for session setup involving less than four network-control entities. This is the case for our test-bed (Figure 4). Indeed, COPS QPS and SIP signaling messages cross only three network control entities: a SIP Proxy, a QPS PEP and a QPS PDP. The test-bed, previously presented, is then used to measure the observed delays introduced by each of our building blocks.

5.1 Scenario

The established test-bed, shown in Figure 4, allows us to set up multiple feasibility and scalability testing scenarios. Demonstrating the feasibility and scalability of our solution consists of analyzing the behavior of SIP and COPS, and verifying that the delay bound defined by the ITU-T is never exceeded. The feasibility testing scenarios reside on measuring this delay on both local and distant configurations.

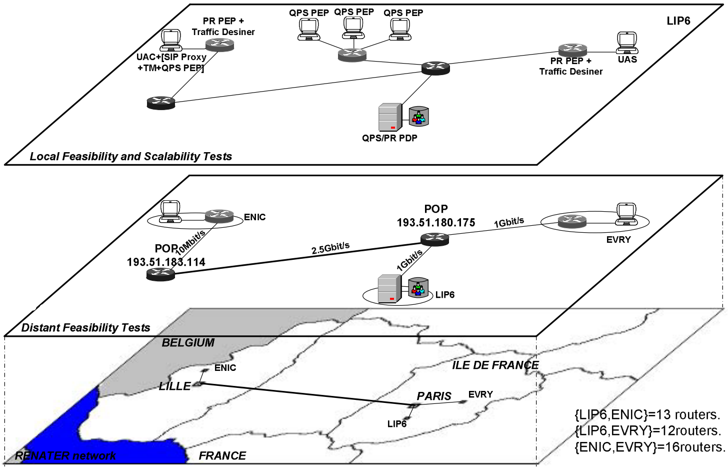


Fig. 4. Test-bed

In the local configuration, all the test-bed components are in the same LAN. In contrast, for the distant feasibility tests, the test-bed components are deported on 3 different LANs interconnected over the public Internet (i.e. over the inter-university IP network called RENATER [16]). Hence:

- The initiator of the multimedia communication (SIP UAC + SIP Proxy + TM + QPS PEP) is connected through an edge router (PR PEP) to the RENATER Network. These components are located in ENIC premises in the city of Lille.
- The responder (UAS) is also connected through an edge router (PR PEP) to the RENATER Network. These components are located in IIE premises in Evry.
- The PDP is located in a third location which is the LIP6 laboratory in Paris.

For the distant feasibility test, the distance, in terms of number of router interconnecting each pair of sites, is also depicted in Figure 4. For this scenario, let's consider an important number of users relying QoS requests. As one SIP proxy is dedicated to each user and not shared, the only bottleneck of our solution becomes the PDP. The scalability testing scenarios consist then of measuring the session establishment delay when the PDP is overloaded. The idea is to measure the effect of increasing the QoS-request rate. This rate is increased by the periodical request generation by three additional terminals hosting a set of virtual PEPs (Figure 4).

Let's note that 30 iterative session-establishment demands are initiated by the SIP UAC in order to compute a statistically acceptable delay estimate. This is realized for each testing scenario.

5.2 Local Feasibility Tests

From Table 1, our attention is focused on the small overall measured delays, compared to the 6s ITU-T bound. Furthermore, the delay's standard deviation is quite low, confirming the accuracy of these experiments. This is not very surprising as these tests are down locally on the same sub-network and no concurrent QoS requests are handled in the same time by the PDP.

Table 1. Local feasibility tests: delay measurements

Delay (ms)	Min	Max	Mean	SD	CV (%)[2]
SIP	409	502	446	25.05	6
TM	2	18	6	3.32	55
PBM	137	424	238	59.81	25
Overall	571	859	690	63.86	9

Table 1 also shows that the simultaneously hold delays for PBM are 50% smaller than those of SIP. This is due to the format and the size of COPS and SIP messages that influence their treatment delays. Indeed, the SIP messages are of text type (said HTTP-like) and need a parser to be interpreted. Quite the opposite, the COPS messages handle objects and are treated by fastest API. To show the size difference between COPS and SIP messages, we can note that the size of COPS REQ is about 356 octets and that the size of SIP INVITE message is about 930 octets.

We can notice finally that the translation module delays are negligible in comparison to the global session establishment delays.

5.3 Distant Feasibility Tests

Let's recall for these tests, the test-bed components are located in three different interconnected LANs throughout the public Internet. The same measures as the previous feasibility test are realized (Table 2).

Table 2. Distant feasibility tests: delay measurements

Delay (ms)	Min	Max	Mean	SD	CV (%)
SIP	441	1429	552	223.33	41
TM	2	22	5	4.47	89
PBM	138	846	282	173,03	61
Overall	638	2099	838	337.22	40

When comparing the results showed in table 2 with the results obtained in the previous section, one can note the significant increase of the maximum delay and the

[2] The coefficient of variation (CV) highlights the relative dispersion of the measured delays.

standard deviation for both SIP and PBM. But, when looking at the minimum and the mean delays, we can remark a little increase compared to the previous test. This latter is not surprising due to the fact that these delays are mainly influenced by the computational overhead rather than the experienced network delays that are minimal. By detailing these results, we note that only 3 SIP and 2 PBM exchanges have a delay more than 700ms and 400ms respectively. These excessive delays are those causing the significant increase of the maximum delay and the standard deviation. They are due to small-transient delay-increase in the network. However, note that the obtained overall delays are not so high; even the maximum is 2.099s and the standard deviation is relatively small. In this way, our solution's overall delays stay clearly bellow the critical limit.

5.4 Scalability Tests

In order to analyze the scalability of our solution, the load of the PDP is progressively increased. In parallel to this increase of load, session establishment demands are initiated. The observed delays for each entity are then measured and analyzed.

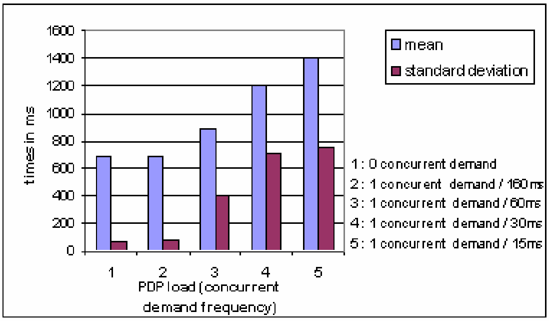


Fig. 5. Overall delays in the scalability testing scenario

Figure 5 summarizes the results obtained in our scalability tests. It highlights the evolution of overall delays obtained for different PDP loads. In this curve, the obtained delays for loads that are smaller than "1 demand / 160ms" are skipped. This is justified by the fact that the corresponding overall-delay linearly grow with a small slope.

From Figure 5, one can note that for rates over 15 demands per second, both mean delay and standard deviation increase drastically. This rate corresponds to the limit after which the PDP load begins to influence clearly on its computational delays.

Detailed delays obtained for the latter test, where 67 demands per second are generated by 3 different PCs emulating 300 connected users, are shown in table 3. This test confirms the non-surprising result related to the SIP operation delays. Indeed, these delays always remain below 500 ms, with a small standard deviation. This is naturally due to the fact that the bottleneck in our solution is not within the SIP part but concerns solely the PDP. However, the PBM operation delays are strongly increased compared to our feasibility tests. One can note that the provisioning and resource allocation delays are the parts that have the highest mean with a very important standard deviation. Indeed, the resource allocation and the provisioning procedures are both carried out by the PDP. The other observed delays remain stable. Let's however precise, that the maximum overall delay, in this latter experiment, is still under the 6s bound. These results consolidate the design philosophy adopted for our solution.

Table 3. Local scalability tests: delay measurements

Delay (ms)	Min	Max	Mean	SD	CV (%)
Policy Enforcement	10	41	28	9.11	33
Provisioning	128	2293	613	639.02	104
Resource Allocation	42	1693	367	479.04	131
SIP	359	458	415	18.57	4
TM	2	18	6	3.32	55
PBM	194	2752	980	744,03	76
Overall	610	3178	1400	746.39	53

A load of 67 demands per second is quit high and the obtained results should give us a good estimate of the overall delays that would certainly be obtained for a realistic worst case. Indeed, this load can correspond to a worst case in a medium size network (eg. a corporation willing to use IP telephony between their different geographical sites). Future improvements of our structure particularly, the use of PDP replication, would lead us to estimate these delays for telecommunication WANs [17].

6 Conclusion

In this paper, we analyzed a novel architecture for multimedia session setup with QoS guarantees. This novel solution deals with the interoperation between SIP functionality and PBM components. Indeed, our proposal is to deport SIP proxy and a QoS Parameter Signaling (QPS) PEP within user's the terminal.

In order to measure the performances of our solution, a complete test-bed has been implemented. It includes all PBM, SIP and integration components. The experiments carried out allow us to highlight the properties of our solution in terms of feasibility and scalability. The feasibility of our solution allows us to find lower overall delay bound for session establishment that is significantly smaller than the upper delay bound recommended by the ITU. Scalability experiments allow us to demonstrate that the bottleneck of our architecture is the PDP. Undeniably, the slope of the PDP operation delay curve begins to drastically increase when the request rate exceeds 15 demands per second. However, tacking all our experiments, this delay always remains below the 6s ITU-T bound.

Note that the test-bed built is realistic for the case of multimedia communications within a corporate Intranet connecting multiple remote sites and aiming to control the resources of the corporate Virtual Private Network (VPN). In this case our solution gives very promising results.

To confirm these results for largest networks, improvements in our architecture (respectively, in our test-bed structure) are needed. These improvements are mainly dealing with PDP replication which will be the issue of our future works.

References

1. L. Bernard and H. Afifi, "QoS dynamique pour applications multimédia basée sur le couplage SIP-COPS et carte à puces," CFIP'03, (Paris, France), Oct. 2003.
2. D. C. Verma, "Policy-Based Networking–Architecture and Algorithms," New Riders Publishing, (Indianapolis, Indiana), Nov. 2000.
3. J. Rosenberg et al., "SIP: Session Initiation Protocol," RFC 3261, Jun. 2002.
4. ITU-T Recommendation No. E.721, "Network grade of service parameters and target values for circuit-switched services in the evolving ISDN," May 1999.
5. J. Boyle, et al, "The COPS (Common Open Policy Service) Protocol," RFC 2748, Jan. 2000.
6. CIM Specifications, "Common Information Model (CIM) Specification Version 2.8," DMTF Policy WG, Aug. 2003.
7. K. Chan, et *al.*, "COPS Usage for Policy Provisioning (COPS-PR)," RFC 3084, Mar. 2001.
8. T.M.T. Nguyen, N. Boukhatem, Y.Ghamri-Doudane,G. Pujolle, "COPS-SLS : A Service Level Negociation Protocol for the Internet," IEEE Communication Magazine, May 2002.
9. Y.Ghamri-Doudane, "QoS Support and Management in Wireless Networks," PhD thesis, University of P. & M. Curie, Nov. 2003.
10. B. Zouari and H. Afifi, et al, "A novel authentification model based on secured IP smart card," IEEE International Conference on Communications, ICC'03, (Anchorage, Alaska), May 2003.
11. J. Hodges et R. Morgan, "Lightweight Directory Access Protocol (v3) : Technical Specification", RFC 3377, Septembre 2002.
12. X. Wang, H. Schulzrinne, D. kandlur, D. Verma, "Measurement and Analysis of LDAP Performance", International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'2000), Santa Clara, CA, pp. 156--165, Janvier 2000.
13. K. Nichols, et al, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, Dec. 1998.
14. JAIN™ SIP API Specification: http://www.jcp.org
15. http://www.qosmos.net/EN/classification.htm
16. http://www.renater.fr/
17. ITU-T Recommendation No. E.500, "Traffic intensity measurement principles," Nov. 1998.

Modeling Soft State Protocols with SDL

Xiaoming Fu and Dieter Hogrefe

Telematics Group, University of Goettingen
{fu, hogrefe}@cs.uni-goettingen.de

Abstract. Soft state provides new services to packet-switching networks by introducing a type of state in the network nodes which is refreshed by periodical messages and otherwise expires. The operations of soft state protocols, which are being designed with ever greater complexity, need to be error-free and deadlock-free to avoid misusing network resources. Thus, verification, formal analysis and validation of these protocols become a vital task. In this paper we utilize formal techniques, specifically, Specification and Description Language (SDL) and Message Sequence Charts (MSCs), for modeling, analysis and validation of various soft state protocols. We propose a general architecture for state management systems and find employing these techniques can help identify and correct possible design errors, which may be caused by informal specifications.

1 Introduction

In communication networks, there is a need to maintain certain information ("*state*") in network nodes, associated with endpoint-generated sessions or calls. For example, ATM switches maintain information about VCs such as bandwidth allocation and VCI/VPI input-output mapping. The state maintained by the network can be categorized as *hard state* and *soft state*. Hard state is installed in nodes upon receipt of a setup message and is removed only upon receipt of an explicit removal message. It is vital for the state initiator to know when the state has been installed or removed, and ensure that installation and removal are performed only once. Furthermore, since hard state remains installed unless explicitly removed, there needs a mechanism to remove orphan state that appears once the state initiator has crashed or departed without removing state. In contrast, "soft state" refers to certain non-permanent state in network nodes which will expire unless refreshed. Since soft state will eventually expire, in principle this approach does not require explicit removal or a mechanism for removing orphan state. Several routing protocols (e.g., BGP, OSPF and RIP) and early Internet signaling protocol (ST-II) use hard state. Once the soft state paradigm was applied to RSVP [1], which allows endpoints to establish QoS reservation state in the network nodes along the path for their end-to-end communications, it has been adopted by many other protocols, such as RTCP, PIM, SIP and CASP [2].

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 289–302, 2005.

By the use of state – either hard state or soft state – inside the network, protocols can provide certain enhanced services for end-to-end communications, such as QoS reservation setup or flow-coupled firewall configurations. Note both hard state and soft state can be installed either in intermediate nodes or end hosts only, or both. Due to the nature of state, unlike other types of protocols, state management protocols often require extremely complex and powerful mechanisms to ensure that the state is perfectly synchronized and up-to-date. With the informal IETF specifications, operations of these protocols tend to be error-prone. For example, a report [3] showed numerous problems or unexpected behaviors in the specification and implementation of TCP, the dominating end-to-end transport protocol which uses hard state in end hosts. With an ever-increasing number of soft state protocols and the increase in their complexity, unfortunately, the risk of design and implementation errors for soft state protocols increases. An example is the "auto-refresh" loop in RSVP, possibly keeping a state alive forever [1]. In general, it is vital to ensure the correctness of state management operations in protocol specifications.

The methods for studying protocol operations and correcting possible flaws can be classified into two basic groups. The natural, empirical ("trial-and-error") approach, is to study them with an actual implementation (or a prototype). Another possibility involves a model-based approach in which protocol behaviors may be studied using a model of the protocol. The empirical approach is only effective for examining standard, ordinary behaviors of a protocol, whereas model-based approaches, based on simulation or analytical models, can be more effective in determining possible errors in the design. Applying the latter approach to state management systems may reduce excessive problems (and costs!) in standards and implementations, unlike the first method of debugging and correcting these specifications. From our experience [4], real soft state protocols can be rather complex and difficult to be analyzed through implementations. This is partly due to the fact that soft state protocols must independently maintain several types of timers and soft states associated with a given end-to-end session in a distributed fashion. Moreover, they are generally specified informally and imprecisely. Therefore, model-based approaches are preferred.

In this paper we employ a model-based approach to study basic functionalities and liveness properties of general soft state systems, using the Specification and Description Language (SDL) [5] and the Message Sequence Charts (MSCs) [6]. As successfully demonstrated in other protocol modeling experiences (e.g., [7]), SDL and MSCs are efficient tools for such tedious tasks. Based on [8], we present a general architecture for state management systems and model such a system with SDL/MSC. Demonstrating the feasibility of applying this approach in modeling soft state protocols and analyzing their basic properties, we advocate a potential way of improving protocol descriptions. We focus on the functionality modeling and validation following [8], including verification of (the absence of) deadlocks and livelocks. Our goal here is not to model a real soft state protocol such as RSVP or CASP, or hard state protocols like ST-II or TCP, but rather to model general soft state protocols in order to capture and verify the essential concepts and general functionalities of interest. There are other, non-functional aspects of these protocols, such as complexity and performance, but they are beyond the scope of this paper.

The rest of this section discusses related works. Given the importance and difficulty in analyzing soft state protocols, Section 2 presents a general architecture covering all variants of soft state protocols, followed by SDL models for representative soft state management (Section 3) and a verification of the models (Section 4). Section 5 summarizes our experiences and outlines future work.

Previous Studies on Soft State Protocols: System designers argue soft state is “better” than hard state, and using soft state the handling of network condition changes is “easy” [9, 10]. However, these claims are more based on intuitive, high-level thoughts and explanations, instead of formal, exhaustive modeling and analysis. In contrast to original expectations, soft state protocols developed and being developed so far are still far from being simple, especially when coupled with multicast sessions or traffic control models.

There are two types of soft state protocols which have been developed so far: end-to-end protocols and hop-by-hop protocols. The former only involves certain types of state in an end-to-end way, without involving any other nodes in between; examples of this type include RTCP and SIP. Hop-by-hop protocols, such as RSVP and CASP, on the other hand, involve state in one or more router(s) in between in addition to state in the communicating ends. For the purpose of demonstration and general discussions of soft state operations, we have chosen to use the latter since it is more representative and comprehensive.

Given the particular importance of soft state protocols, recent studies have looked at issues regarding their modeling and analysis. Raman and McCanne [9] presented a model for the soft state notion based on Jackson queueing networks; a performance study of hard state and soft state signaling protocols was performed by Ji *et al.* [8]. Unfortunately, these studies lack more detailed formal modeling and validation. Bradley *et al.* [11] studied the correctness and interoperability issues with the HTTP protocol, and established that multi-stage interactions between the HTTP server and clients can be stateful and error-prone; they then verified these behaviors by using a formal checking tool SPIN [12]. However, their study is limited to application-layer hard state management between two endpoints and does not consider the soft state paradigm for packet-switching networks (which may involve multi-hop behaviors of state management systems).

A Brief Introduction to SDL and MSC: An SDL system is divided into building blocks that communicate using channels, whereas blocks are further composed of processes. Processes (within a block) are connected using signal routes. Each process is an extended finite state machine, which has its own infinite queue and is assumed to operate independently from all other processes. MSCs are another valuable description technique for visualizing and specifying inter-system, asynchronous component interaction. MSCs’ strength lies in their ability to describe communication between cooperating processes. Each process is represented as an identifier and has a process life line that extends downward.

There are arrows representing messages passed from a sending to a receiving process. Messages not starting or ending at a process life line are exchanged with users, be they human or mechanical (the "*environment*"). Detailed descriptions of SDL and MSC can be found in [5, 6]. Modern SDL development tools like Telelogic Tau also support verification and validation based on developed models.

2 A General Architecture of State Management Systems

In contrast to hard state (HS) protocols, soft state (SS) protocols can be further classified into 4 variants: "pure" soft state (pSS), soft state with explicit removal (SS+ER), soft state with reliable trigger (SS+RT) and soft state with reliable trigger/removal (SS+RTR) [8]. In this section, we present a simple abstraction and typical operations for all these state management protocols (both soft state and hard state), as well as possible problems that may occur in their operations.

We begin our modeling with a generic architecture for state management systems as shown in Fig. 1, which covers two services and one protocol as below:

- The state message transport (ST) service, which transmits state messages over the lossy channel.
- The state management service (SMP service), which is the service rendered by the state management protocol to the state application (SA).
- The state management protocol (SMP) managing state in network nodes.

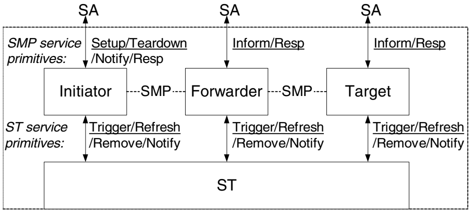

SA
SA
SA
SMP service primitives:
Setup/Teardown /Notify/Resp
Inform/Resp
Inform/Resp
Initiator
SMP
Forwarder
SMP
Target
ST service primitives:
Trigger/Refresh /Remove/Notify
Trigger/Refresh /Remove/Notify
Trigger/Refresh /Remove/Notify
ST

Fig. 1. A general architecture of state management systems

Expected Behavior of State Management Protocols: Behaviors of a state management protocol are determined by timers and state messages used by the three types of protocol entities, namely the state initiator, forwarder and target. An important feature of soft state systems is timers. In a soft state system, there can be three types of timers dealing with state management: the *state timer* which will expire the state unless refreshed, the *refresh timer* which triggers periodical refreshes, and the *retransmission timer* which triggers periodical retransmission of trigger or removal messages (SS+RT or SS+RTR). In HS systems there are only retransmission timers for state trigger and removal messages, while state timers and refresh timers are necessary for SS systems for operating

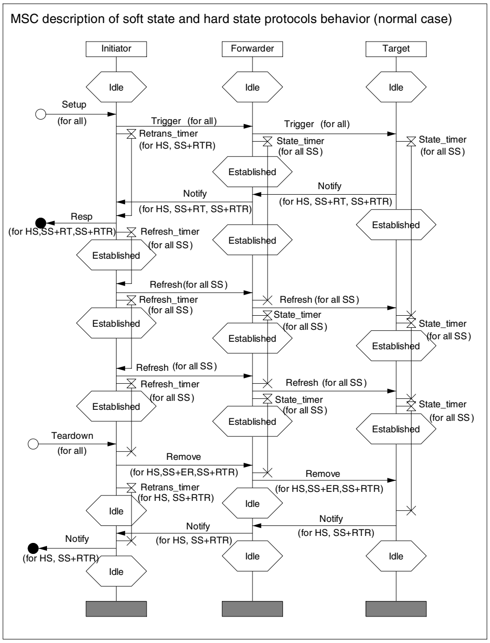


Fig. 2. MSCs for typical operations in various state management protocols

state refresh messages. The existence or absence of these timers and different operations for state messages in a state management protocol determines which protocol type it belongs to.

The MSCs are shown in Fig. 2 to illustrate various sate management protocols. The communications between the three types of entities are realized through several service primitives (Setup, Trigger, Teardown, and optionally, Refresh, Resp, Notify and Remove). For simplicity purposes, we omit the Inform and Resp primitives since they generally do not change state information.

The protocol communication takes place in 3 possible, distinct phases:

- *State Setup:* This phase is initialized by SA in the Initiator with a *Setup.* Initiator can thereafter issue a *Trigger* towards Target. Upon the receipt of Trigger, every Forwarder creates a state (which is associated with a state timer for soft state protocols), and then forwards Trigger on. When Target receives Trigger, it creates a state (which is associated with a state timer for soft state protocols). If HS, SS+RT or SS+RTR is used, additionally Target issues back a *Notify* to Initiator when it receives a Trigger. In this case, Initiator starts a retransmission timer before it issues Trigger. If it does not receive a Notify after the Retransmission timer expires, Trigger is transmitted again. After repeating certain times, retransmission stops and Initiator becomes inactive again.

- *State Maintenance:* This phase is only used in soft state protocols. Upon the expiration of the refresh timer, Initiator sends a *Refresh*[1] towards Target and restarts the timer. Any Forwarder or the Target receives the Refresh checks whether corresponding state already exists: if yes, refreshes the state timer, otherwise recovers a state together with a state timer. If it is not the Target, the node forwards the Refresh onwards. If no Refresh is received in a Forwarder or Target before the state timer expires, state will be removed.
- *State Teardown:* In pSS or SS+RT, there is no such phase; Initiator remains inactive and states will expire in all the other nodes when their state timers time out. In other state management protocols, a *Remove* is issued by the Initiator towards the Target to remove all state and associated timers (should they exist). Additionally, if HS or SS+RTR is used, after Target receives Remove and removes its state (and timers), a *Notify* is sent back to the Initiator. Initiator in a HS or SS+RTR system retransmits Remove (no more than a maximal retrial counter) if no Notify is received within a given time.

Possible Problems in State Management Operations: Because the above general model captures the key concepts and operations of all the 5 possible types of state management protocols, we believe it can serve as the basis for studying actual behaviors of real state management systems. The most obvious problem that occurs in state management protocols is failure to install or remove state correctly. In addition, there are timing considerations to be taken account, since a protocol of this kind needs to be able to react to timer events and receipts of state messages appropriately. An installed state can become invalid either due to the receipt of a removal message or when the state timer expires. The latter can occur when a state refresh or notify message gets lost during its transmission, or when the state initiator crashes.

There are other potential problems with state management protocols, e.g., infinite state management loops, i.e., state messages enter a transmission circle somewhere between an initiator and a target. This can be either an endless loop consisting of a set of interacting states which cannot progress towards a next expected behavior (livelock), or a state without possibilities to enter another state (deadlock). A deadlock is most often caused by two processes waiting for a message from each other; the result is that both wait and nothing happens. In pSS, such loops are theoretically impossible, as it only allows refresh and expiration operations for a state, and there is no resource contention. However, we cannot exclude this possibility in more comprehensive state management protocols, due to more complex synchronization and/or notification mechanisms.

These are very undesirable since there appears to be a valid state management behavior in any individual node, but in reality it cannot further deliver other desired messages or jump out of certain running state(s). For example, the following operation could be possible in the original description of SS+RT [8].

[1] Note some SS protocols (e.g., RSVP) allow initiated by intermediate nodes; here we limit refreshes to be originated from Initiator for simplicity.

Example 1. A deadlock behavior in SS+RT can be briefly explained as follows:

1. Initiator initially sends a Trigger to Target through Forwarder. The link between Forwarder and Target then suddenly goes down.
2. Trigger is lost before reaching Target, and Initiator cannot receive a desired Notify which acknowledges the success of state installation along the path.
3. After retransmission timer expires, Initiator resends a Trigger.
4. Again the Trigger is lost, and Initiator still thinks it is just due to random loss and retransmits the Trigger.
5. If there is no specification about which conditions to stop sending Triggers, the result is a deadlock in which case Initiator is waiting for a Notify after sending out a Trigger while Responder is waiting for a Trigger forever.

This error is somewhat easy to detect and fix. However, sometimes such flaws can be very subtle, so that even senior designers cannot detect them in protocol specifications, particularly in IETF informal, text-based specifications for complex operations of state management protocols, which were designed typically without formal modeling, validation and verification. For example, the inadvert synchronization problem [13] was noticed as an important issue for periodical soft state systems. In real specifications of some soft state protocols, for example RTCP and RSVP, designers have tried to avoid this problem by setting the refresh timers to be varied randomly (e.g., over the range [0.5, 1.5] times the calculated interval). However, true randomness in a real implementation is hard to achieve and moreover, as these intervals are sometimes not mandatory requirements in protocol specifications (e.g., as a “should” requirement in RSVP), typically default fixed values (30 seconds in case of RSVP) are used instead in practice for implementation simplicity. All these contribute to the potential synchronization problem of soft state management.

3 Modeling Soft State Protocols with SDL

Key Modeling Issues and the System Model: Based on the above general architecture, we chose the most comprehensive state management protocol, SS+RTR in a hop-hop manner, as the target for modeling. Other variants can be easily derived from this model by removing certain messages, timers, state transitions and/or behaviors.

The address of each node is represented by a pre-assigned integer, 1, 2, 3, respectively. The state message content is also simplified as a simple character string. There are further issues vital for the modeling process:

- How do we model ST, so as to allow state messages (generally from Initiator to Target) be visible for Forwarder?
- How do we model the lossy channel, which can be of a given loss rate?
- How do we model the duration (start and end) of a session state? This also naturally reflects the system model, namely which information should be made inside the system, and which needs to be put in the environments.

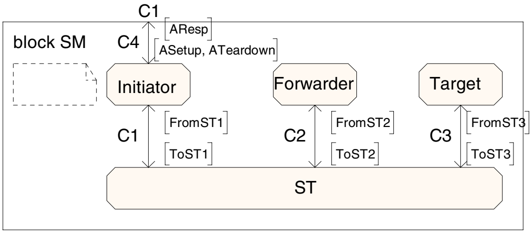


Fig. 3. SDL blocks of the SMP system

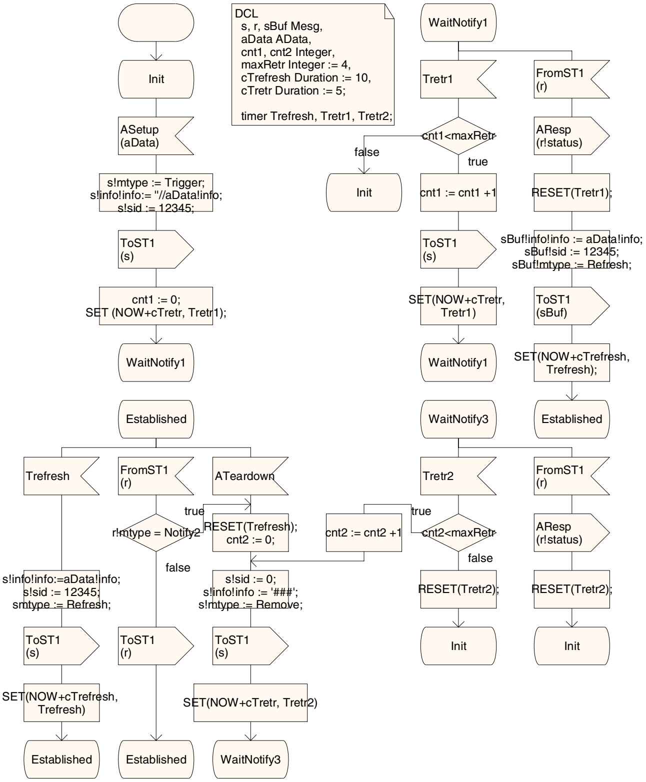


Fig. 4. SDL model for Initiator in the SMP system (SS+RTR)

Fig. 3 shows the SMP system model for SS+RTR. We assume the SMP system model to be composed of 3 nodes: Initiator, Forwarder and Target, each represented by a process. Furthermore, an additional process ST is used to transmit SS messages from Initiator towards Target, or reverse. These 4 processes form a single block *SM* of the system.

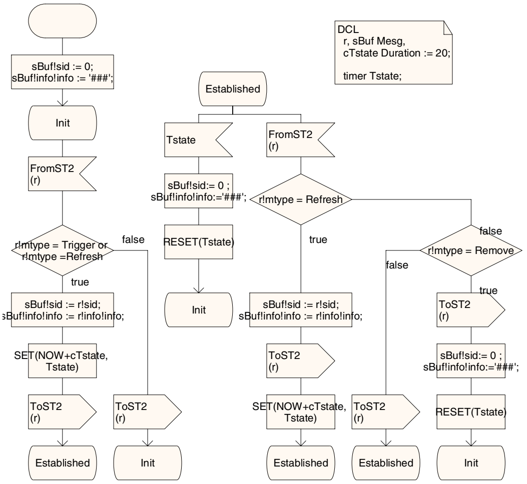


Fig. 5. SDL model for Forwarder in the SMP system (SS+RTR)

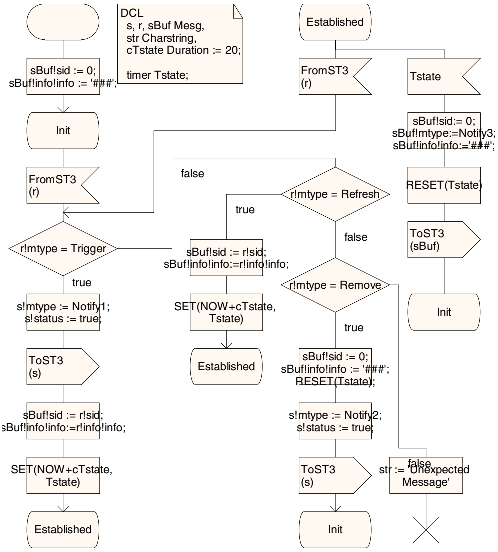


Fig. 6. SDL model for Target (SS+RTR)

ST: ST is modeled as Fig. 4 and can be used for all SMP variants. We use random Abstract Data Type (ADT) to simulate a given loss rate of the link. Note the transport service should determine the direction (forward or backward) according to the type of received messages It is easy to extend to form more comprehensive STs e.g., those having different loss rates in different links.

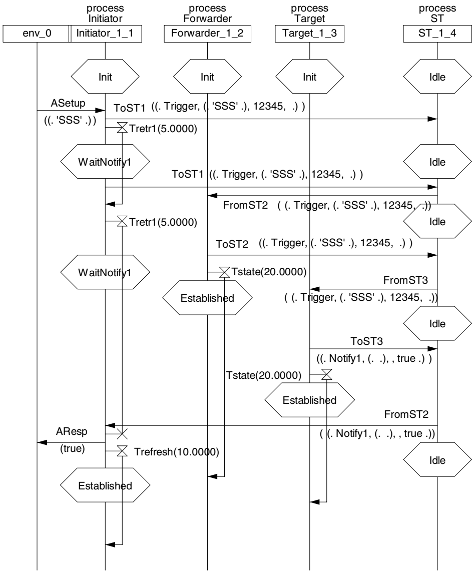


Fig. 7. MSCs for state setup phase (SS+RTR)

SMP Entities: The Initiator process (Fig. 5) communicates with environments through an SA-SMP interface. When it receives an ASetup with certain state information data, it assigns a new session identifier (sid) and installs a soft state locally, before going through the state setup and maintenance phases. When Initiator receives an ATeardown from the SA-SMP interface, it enters the teardown phase. For simplicity sid is currently set as a fixed value. To avoid confusion different notification messages have to be identified: Notify1 for notifying the success of state setup, Notify2 for notifying state timer expiration in Target, and Notify3 for notifying the success of state teardown.

The Forwarder process (Fig. 6) first listens to ST input. If a Trigger or Refresh message arrives, it installs a soft state locally and forwards the message on. Expiration of a local state in Forwarder only removes itself.

The Target process (Fig. 7) installs and (if it should exist) refreshes soft state locally upon receipt of a Trigger or Refresh message. When receiving a Trigger or Remove message, Target needs to notify Initiator about it. Expiration of a local state in Target also accompanies a Notify2 to trigger a teardown.

4 Model Verification and Validation

Since state management protocols involve distributed timers and message interaction, their correctness is hard to verify using an informal description. We use the Tau integrated package SDT 4.4 which includes support for SPIN, to verify

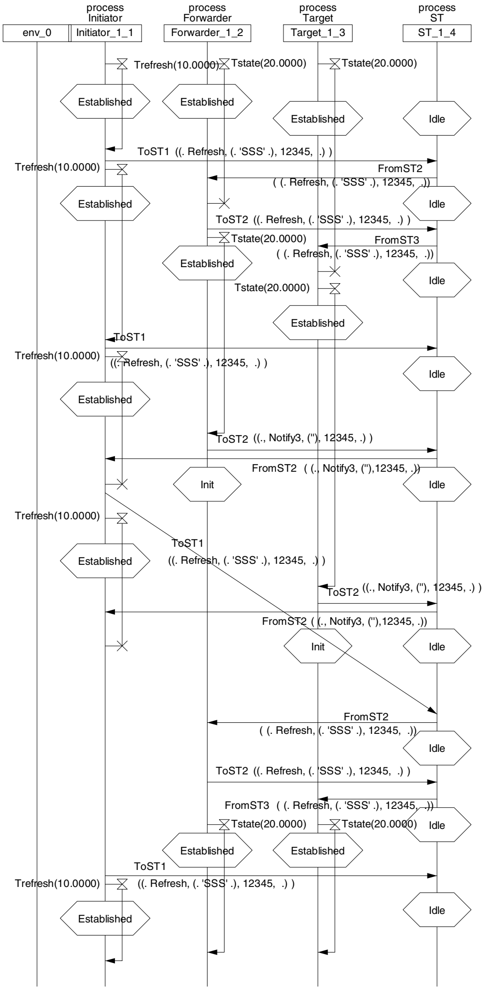

process Initiator
process Forwarder
process Target
process ST
env_0
Initiator_1_1
Forwarder_1_2
Target_1_3
ST_1_4
Trefresh(10.0000)
Tstate(20.0000)
Tstate(20.0000)
Established
Established
Established
Idle
ToST1 ((. Refresh, (. 'SSS' .), 12345, .))
Trefresh(10.0000)
FromST2
((. Refresh, (. 'SSS' .), 12345, .))
Established
Idle
ToST2 ((. Refresh, (. 'SSS' .), 12345, .))
Tstate(20.0000)
FromST3
((. Refresh, (. 'SSS' .), 12345, .))
Established
Idle
Tstate(20.0000)
Established
ToST1
Trefresh(10.0000)
((. Refresh, (. 'SSS' .), 12345, .))
Idle
Established
ToST2 ((., Notify3, (''), 12345, .))
FromST2 ((., Notify3, (''),12345, .))
Init
Idle
Trefresh(10.0000)
ToST1
((. Refresh, (. 'SSS' .), 12345, .))
Established
ToST2 ((., Notify3, (''), 12345, .))
FromST2 ((., Notify3, (''),12345, .))
Init
Idle
FromST2
((. Refresh, (. 'SSS' .), 12345, .))
Idle
ToST2 ((. Refresh, (. 'SSS' .), 12345, .))
FromST3 ((. Refresh, (. 'SSS' .), 12345, .))
Tstate(20.0000)
Tstate(20.0000)
Idle
Established
Established
ToST1
Trefresh(10.0000)
((. Refresh, (. 'SSS' .), 12345, .))
Idle
Established

Fig. 8. MSCs for state maintenance phase (SS+RTR)

the SDL model of the SS+RTR protocol against deadlocks, unspecified receptions, livelocks and unreachable states. In order to verify the model, we have chosen the following scenario case study to validate our design against some of the specific properties of the modeled SS+RTR protocol:

1. Establish a session between Initiator and Target.
2. Stop the state message transmission between Forwarder and Target.
3. Change the ST loss rate between Forwarder and Target to different values between 0 and 1.
4. Stop the Target process.
5. Let Initiator teardown the session.

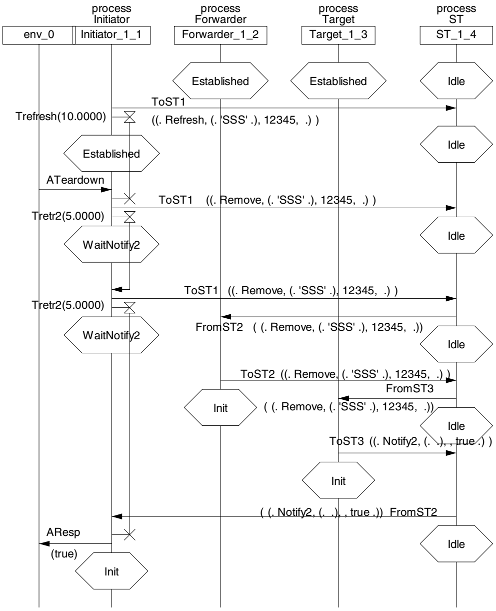

process Initiator
process Forwarder
process Target
process ST
env_0
Initiator_1_1
Forwarder_1_2
Target_1_3
ST_1_4
Established
Established
Idle
ToST1
Trefresh(10.0000)
((. Refresh, (. 'SSS' .), 12345, .))
Established
Idle
ATeardown
ToST1 ((. Remove, (. 'SSS' .), 12345, .))
Tretr2(5.0000)
Idle
WaitNotify2
ToST1 ((. Remove, (. 'SSS' .), 12345, .))
Tretr2(5.0000)
FromST2 ((. Remove, (. 'SSS' .), 12345, .))
WaitNotify2
Idle
ToST2 ((. Remove, (. 'SSS' .), 12345, .))
FromST3
Init
((. Remove, (. 'SSS' .), 12345, .))
Idle
ToST3 ((. Notify2, (. .), , true .))
Init
((. Notify2, (. .), , true .)) FromST2
AResp
(true)
Idle
Init

Fig. 9. MSCs for state teardown phase (SS+RTR)

With the above scenario, we have covered all ST service primitives and SMP service primitives as well as all important scenarios, but not all possible scenarios. Therefore, after checking the scenario, we have used Tau validator to validate all possible walk algorithms.

We generated a number of MSCs for this scenario case study to check the protocol functionality at each stage of the simulation. For the SDL models designed in Section 3, MSCs shown in Fig. 7-9 help us to identify the representative protocol behaviors: 1) Fig. 7 represents the message interactions for state setup phase (SS+RTR) upon the receipt of an external trigger (ASetup) is received by the system, completed by a response indicating successful state establishment from the system to the environment (AResp); 2) after that, the system goes into the internal state maintenance phase without interactions with the environment (shown in Fig. 8); 3) upon the receipt of an external trigger for state teardown (ATeardown), the system enters into the state teardown phase (Fig. 9). Through a comparison with the general description, we are able to refine the abstract system and make more concrete descriptions.

Through this procedure, we have found some modeling errors as well as flaws in the original description, including the missing default behavior for some message types, and unreachable states. We have found that an imprecise informal specification can result in deadlocks and livelocks (for example, the retransmission counters and refresh durations – either explicit or implicit – are missing in many text-based IETF specifications); through verification and validation of the SDL models, these could be avoided. As an example we corrected a flaw in the original description by adding description on retransmission counters.

Some other results show that in the first version of the models Notify1, Notify2 and Notify3 messages received by Forwarder were not processed but consumed; they need to be sent back to ST. Also, misused values of timers can exclude Initiator from entering the Established state. We also come to the following conclusion: by adding reliable trigger and explicit removal to soft state protocols, the usage of state (reflected as network resources especially memories) can be more efficient. This can be explained as, for example for the reliable explicit removal case, if a user tries to remove a state, but the teardown message is lost during transmission, the state will remain in place until it times out after a relatively long time. Since state typically implies enhanced services for end-to-end communications, maintaining a state incurs costs, thus users must pay for the extra time that has been spent waiting for the state expiration.

With the developed model, we have verified these aspects against the description of SS+RTR, and improved upon it.

5 Summary and Future Work

Given the fundamental importance of soft state protocols, it is vital that their behaviors are specified correctly. We have generalized and modeled behaviors for different variants of soft state communications with the aid of formal techniques SDL and MSC. On the other hand, we found that a weakness concerning inaccurate timing in the tool; a more realistic performance evaluation has to rely on tools with better real-time support. Two types of error sources are identified during the modeling processes: one due to technical flaws in the specifications and the other due to our modeling errors (which can also be caused by misunderstanding of imprecise specifications). Nevertheless, formal techniques have turned out to be of great help for efficient designing and engineering of soft state communication models, and in particular functional behaviors (through checking for possible deadlocks and livelocks). Currently we are modeling two realistic soft state protocols, CASP and RSVP. Based on that we will further study how soft state protocols handle node failure, mobility and route changes using formal techniques.

References

1. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (1997)
2. Schulzrinne, H., Tschofenig, H., Fu, X., McDonald, A.: CASP – Cross-Application Signaling Protocol. Internet draft, work in progress (2003)
3. Paxson, V., Allman, M., et al.: Known TCP Implementation Problems. RFC 2525 (1999)

4. Fu, X., Hogrefe, D., Willert, S.: Implementation and Evaluation of the Cross-Application Signaling Protocol (CASP). In: Proc. of ICNP 2004, Berlin, Germany (2004)
5. ITU-T Recommendation Z.100 – Specification and Description Language (SDL). (1999)
6. ITU-T Recommendation Z.120 – Message Sequence Chart (MSC). (1999)
7. Schaible, P., Gotzhein, R.: View-based Animation of Communication Protocols in Design and in Operation. Computer Networks **40** (2002) 621–638
8. Ji, P., Ge, Z., Kurose, J., Towsley, D.: A Comparison of Hard-state and Soft-state Signaling Protocols. In: Proc. of SIGCOMM 2003, Karlsruhe, Germany (2003)
9. Raman, S., McCanne, S.: A Model, Analysis, and Protocol Framework for Soft State-based Communication. In: Proc. of SIGCOMM 1999, Cambridge, MA (1999)
10. Sharma, P., Estrin, D., Floyd, S., Jacobson, V.: Scalable Timers for Soft State Protocols. In: Proc. of INFOCOM'97, Kobe, Japan (1997)
11. Bradley, A., Bestavros, A., Kfoury, A.: Safe Composition of Web Communication Protocols for Extensible Edge Services. In: Proc. of Workshop on Web Content Caching and Distribution (WCW), Boulder, Colorado (2002)
12. Holzmann, G.: The Model Checker SPIN. IEEE Trans. on Softw. Engineering **23** (1997) 1–17
13. Floyd, S., Jacobson, V.: The Synchronization of Periodic Routing Messages. IEEE/ACM Trans. on Networking **2** (1994) 122–136

Inferring Traffic Burstiness by Sampling the Buffer Occupancy

Michel Mandjes[1] and Remco van de Meent[2]

[1] CWI, Amsterdam, Netherlands
michel@cwi.nl
[2] Univ. of Twente, Enschede, Netherlands
r.vandemeent@utwente.nl

Abstract. Common practice to determine the required bandwidth capacity for a network link is to measure the 5 minute average link load, and then add a safety margin to cater for the effect of burstiness on small time-scales. Because of the substantial measurement efforts required to determine the burstiness, network managers often rely on rules of thumb to find the safety margin, e.g. 'mean plus 50%'. In this paper we propose a novel method to accurately determine the burstiness of traffic on small time-scales, *without* requiring measurements on such small time-scales. Our method is based on coarse-grained polling of the occupancy of a buffer in front of the link, from which the burstiness on small time-scales is inferred. We provide the theoretical foundations of our approach, and a validation through both simulation using synthetic traffic as well as real network traffic taken from various operational networks. It turns out that using our approach, it is possible to accurately determine burstiness on small time-scales (for instance 10 ms), by sampling the buffer occupancy (for instance) every second.

Keywords: Network design and capacity planning, queueing network models.

1 Introduction

Provisioning of network resources addresses the interrelationship between: (i) offered traffic (in terms of both average load and burstiness), (ii) desired level of performance, and (iii) the required capacity. Generally, more capacity is needed when offered load and burstiness increase, or when the performance criterion becomes more stringent. To operate a network in a viable way, provisioning procedures balancing (i), (ii), and (iii) are required: scarce provisioning inevitably leads to performance degradation, whereas (too much) over-provisioning results in a waste of resources.

Provisioning procedures are commonly based on rules of thumb. Considering a time window in which network traffic can be assumed stationary, one often uses rules of the type 'the mean traffic rate, plus a margin of 50%'. Obviously, such a fixed margin is not universally applicable. This motivates the use of formulae of the type

$$C = M + \alpha\sqrt{V} \tag{1}$$

for mean load M, some factor α (reflecting the performance target, which is mostly determined by the applications involved), and a standard-error term $\sqrt{V}$ to account for

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 303–315, 2005.

traffic fluctuations, see for instance [1]. Hence, the size of the 'safety margin' is affected by the chosen performance target, and the burstiness of the offered traffic. From (1) we conclude that provisioning (for a given performance target) requires knowledge of both M and V.

The mean traffic rate M can be determined by standard coarse-grained traffic measurements. A common way is to poll Interfaces Group MIB counters via the Simple Network Management Protocol (SNMP) every 5 minutes; this yields the total amount of traffic sent through the network interface over this time-interval. Determining the burstiness V, on the other hand, is more involved, and is essentially the subject of this paper.

Let $A(t)$ denote the traffic offered over a window of length $t > 0$, and the function $V(t) := \mathrm{Var}A(t)$. Then the analysis in this paper indicates that there is some 'dominant time-scale' T, such that the (burstiness) V in formula (1) corresponds to $V(T)$; hence it is this $V(T)$ that needs to be estimated. Note that the time-scale T is usually relatively small (for instance in the order of 100 ms, and often even considerably less). Clearly, $V(T)$ can be estimated directly by doing measurements on time-scale T, and computing the sample variance. However, it is hard to do accurate measurements on these small time-scales; they are hardly feasible through SNMP. This motivates the need for alternative, 'cheap' measurement techniques for determining the traffic variance on small time-scales.

Contribution. As argued above, it is of crucial interest to develop methods for efficiently and accurately estimating $V(T)$ for small time-scales T. The main contribution of this paper is that we propose an alternative for 'direct estimation' (i.e., by doing measurements on the time-scale of interest). In our approach the (i) *buffer occupancy is polled* on a regular basis (for instance every 10 seconds), and (ii) subsequently we 'invert' the resulting (estimated) buffer content distribution to the variance function $V(\cdot)$. Importantly, this approach eliminates the need for traffic measurements on small time-scales. In this sense, we remark that our proposed procedure is rather counterintuitive: without doing measurements on time-scale T, we are still able to accurately estimate $V(T)$. In fact, one of the attractive features of our 'inversion procedure' is that it yields the *entire* 'variance curve' $V(\cdot)$ (of course up to some finite horizon), rather than just $V(T)$ for some pre-specified T.

Approach and organization. The variance estimation technique proposed in this paper relies on the assumption that traffic is (fairly) Gaussian: for any $t \geq 0$, the amount of traffic offered in a time window of length t is accurately described by a normal distribution, parameterized by a mean Mt and variance $V(t) := \mathrm{Var}A(t)$. This Gaussianity was observed in various measurement studies. Kilpi and Norros [2] have statistically verified that the use of Gaussian traffic models is justified as long as the aggregation is sufficiently large (both in time and number of flows), due to Central Limit type of arguments. Importantly, Gaussian models cover long-range dependent processes, such as fractional Brownian motion (fBm); traffic measurements in the 1990s showed that in various situations this fBm model accurately models network traffic, see, e.g., [3].

Section 2 presents preliminaries on Gaussian queues. Particular attention is paid to provisioning formulae (for both buffer and bandwidth), in line with (1). The provisioning formula motivate the need for methods to estimate the variance function $V(\cdot)$, i.e., $V(t)$ as a function of the interval length t. The formulae for Gaussian queues lead to our efficient method for estimating the $V(\cdot)$, that is explained in Sect. 3; importantly, we derive an 'inversion formula' that yields $V(\cdot)$ from the empirically determined buffer content distribution. In Sect. 4 we describe the inversion procedure, and demonstrate it by using synthetic traffic (in this case fBm traffic).

In the inversion procedure, we have identified three sources of possible errors, viz.: the (large-deviations) asymptotics give an *approximation* of the overflow probability (rather than an exact formula), the buffer content distribution is *estimated* through the buffer polling procedure, and we *assume* that traffic is (accurately approximated by) Gaussian. In Sect. 5 we present a detailed, quantitative study of the impact of each of these errors on the resulting estimation of the variance.

To investigate the applicability of the procedure in practice, we have performed extensive numerical experiments with real data from various real-life settings. These networks have different access technologies and link speeds, and different applications and user populations. We present and discuss the results of this experimental validation in Sect. 6.

Section 7 presents a number of reflections of the feasibility of implementing our estimation procedure, and concludes that there are no conceptual impediments. Section 8 concludes, and and lists a number of possible directions for future work.

2 Gaussian Queues – Motivation

In this section we review some basic principles of Gaussian traffic, and recapitulate the main fundamental (large-deviations) theory for queues with Gaussian input. Then we derive, for this Gaussian setting, a number of dimensioning rules. These formulae motivate the need for estimating specific traffic characteristics, viz., the mean rate M and the variance function $V(\cdot)$, cf. provisioning rule (1).

Preliminaries on Gaussian queues – many-sources asymptotics. Consider n independent, statistically identical *Gaussian* sources. It is assumed that the traffic pattern generated by an individual source corresponds to a Gaussian process with stationary increments. This type of sources is characterized by their mean traffic rate μ, and their variance function $v(t)$, for $t \geq 0$. With $A_i(t)$ denoting the amount of traffic generated by the ith source in an interval of length $t \geq 0$, then $\mathbb{E}A_i(t) = \mu t$, and $\mathrm{Var}A_i(t) = v(t)$.

Now suppose that the n sources feed into a queue with capacity C, and apply the scaling $C \equiv nc$. It is well-known that the stationary queue length, say Q_n has the same distribution as the maximum of the corresponding 'free-buffer process': $Q_n \overset{\mathrm{d}}{=} \sup_{t>0}\left(\sum_{i=1}^{n} A_i(-t,0) - nct\right)$. The following fundamental result is found in, e.g., [4]:

Lemma 1. *Suppose that there is an $\alpha < 2$ such that $v(t)/t^{\alpha} \to 0$ for $t \to \infty$. Then, for any $b > 0$, and $c > \mu$:*

$$I(b) := -\lim_{n\to\infty}(1/n)\log\Pr(Q_n \geq nb) = \inf_{t>0}(b + (c-\mu)t)^2/(2v(t)) \ .$$

The above result holds for the system 'scaled by n', but gives rise to an approximation for the 'unscaled' situation, see also for instance [4], [5–Eq. 3]. With $B \equiv nb$, consider the probability that the buffer content Q exceeds B. Denote $M \equiv n\mu$ as the aggregate mean, and $V(t) \equiv nv(t)$ as the aggregate variance.

Approximation 2. *For any $B > 0$, and $C > M$,*

$$\Pr(Q > B) \approx \exp\left(-\inf_{t>0}\ (B + (C - M)t)^2/(2V(t))\right) \ . \tag{2}$$

Provisioning formulae. One of the major tasks in network management is the provisioning of resources: choose the link rate and/or buffer size such that some pre-specified performance criterion is met. The above approximation formula (2) provides us with a tool for developing such provisioning rules. For several performance criteria, we derive the corresponding rules.

Link provisioning of an unbuffered resource. Suppose the goal is to provision the link such that the probability of exceeding the capacity C for a period of length T is smaller than ϵ. Hence we have to find the smallest $C = C(T, \epsilon)$ such that: $\exp\left(-((C - M)T)^2/(2V(T))\right) \le \epsilon$, cf. (2). It is readily checked that this yields, with $\delta := \sqrt{-2\log\epsilon}$,

$$C(T, \epsilon) = M + \delta/T \cdot \sqrt{V(T)} \ . \tag{3}$$

Notice that $C(T, \epsilon)$ decreases in ϵ, as expected: the less stringent the performance target, the less bandwidth is needed.

Link provisioning of a buffered resource. In the setting of provisioning rule (3) we considered an unbuffered resource. In practice, however, network elements are often equipped with a queue, to absorb traffic rate fluctuations. If the router has a queue of size B, and suppose we wish to provision the capacity, we have to find the minimal $C = C(\epsilon)$ such that (2) is below ϵ. Hence we are searching for:

$$\min\left\{C \mid \forall t > 0 : \exp\left(-(B + (C - M)t)^2/(2V(t))\right) \le \epsilon\right\} \ .$$

After rearranging terms, we find:

$$C(\epsilon) = M + \inf_{t>0}\left(\delta/t \cdot \sqrt{V(t)} - B/t\right) \ . \tag{4}$$

Again, the bandwidth required decreases in ϵ. Moreover, it also decreases in B: the larger the queue, the better traffic fluctuations can be absorbed by the buffer, and hence less link capacity is needed.

Buffer provisioning. Similarly, we can determine the minimum required buffer $B = B(\epsilon)$:

$$B(\epsilon) = \inf_{t>0}\left(\delta\sqrt{V(t)} - (C - M)t\right) \ . \tag{5}$$

Example 3. *fBm. Motivated by several measurements studies [3], we focus here on one of the key models in current traffic theory, namely fractional Brownian motion input, i.e., Gaussian traffic with $V(t) = \sigma^2 t^{2H}$ – take for simplicity $\sigma = 1$. $H \in (0, 1)$ is the so-called Hurst parameter; for network traffic a typical value is 0.7-0.8. Straightforward computations give for (3):*

$$C(T, \epsilon) = M + \delta/T^{1-H} \ .$$

When computing $C(\epsilon)$ in (4), the optimizing t is given by $B^{1/H}(1-H)^{-1/H}\delta^{-1/H}$, yielding:

$$C(\epsilon) = M + \delta^{1/H}\left((1-H)/B\right)^{1/H-1} H \ .$$

In buffer provisioning rule (5) the optimizing t is given by $(\delta H)^{1/(1-H)}(C-M)^{-1/(1-H)}$, such that:

$$B(\epsilon) = \left(\delta H/(C-M)^H\right)^{1/(1-H)} \cdot (1-H)/H \ .$$

The main conclusion from this section is that the above provisioning formulae (3), (4), and (5) indicate that it is of crucial importance to have accurate estimates of the average traffic rate M, as well as the variance curve $V(\cdot)$ (i.e., $V(t)$ as a function of $t \geq 0$); having these at our disposal, we can find the required bandwidth capacity or buffer size. As estimating M is straightforward, the next sections concentrate on efficient methods for estimating the variance curve $V(\cdot)$.

3 Derivation of the Inversion Formula

As mentioned in the introduction, the mean traffic rate M can be determined by standard coarse-grained traffic measurements. It is clear that determining the variance curve $V(\cdot)$ is more involved. The standard way to estimate $V(T)$ (for some interval length T) is what we refer to as the 'direct approach'. This method is based on traffic measurements for disjoint intervals of length T, and just computes their sample variance. It is noted that the convergence of this estimator could be rather slow when traffic is long-range dependent [6–Ch. I], but the approach has two other significant drawbacks:

1. When measuring traffic using windows of size T, it is clearly possible to estimate $V(T)$, $V(2T)$, $V(3T)$, etc. However, these measurements obviously do not give any information on $V(\cdot)$ on time-scales *smaller* than T. Hence, to estimate $V(T)$ measurements should be done at granularity T or less; T is typically rather small. This evidently leads to a substantial measurement effort.
2. The provisioning formulae (4) and (5) require knowledge of the *entire* variance function $V(\cdot)$, whereas the direct approach described above just yields an estimate of $V(T)$ on a pre-specified time-scale T. Therefore, a method that estimates the entire curve $V(\cdot)$ is preferred.

This section presents a powerful alternative to the direct approach; we refer to it as the *inversion approach*, as it 'inverts' the buffer content distribution to the variance curve. This inversion approach overcomes the problems identified above. We rely on the many-sources framework of Sect. 2.

Define the 'most likely epoch of overflow' for a given buffer value $b > 0$: $t_b := \arg\inf_{t>0}(b + (c-\mu)t)^2/(2v(t))$; note that t_b is not necessarily unique. Define the set $\mathcal{T}$ as follows: $\mathcal{T} := \{t > 0 \mid \exists b > 0 : t = t_b\}$. The following theorem gives, for any $t > 0$, an upper bound on the variance $v(t)$, for given $I(b)$, and presents conditions under which this upper bound is tight.

Theorem 4. *(i) For any $t > 0$, $v(t) \leq \inf_{b>0}(b + (c-\mu)t)^2/(2I(b))$;*

(ii) There is equality *for all $t \in \mathcal{T}$;*

(iii) If $2v(t)/v'(t) - t$ grows from 0 to ∞ when t grows from 0 to ∞, then $\mathcal{T} = (0, \infty)$.

Proof. Clearly, due to Lemma 1, for all $b > 0$ and $t > 0$, we have that $I(b) \leq (b + (c-\mu)t)^2/(2v(t))$, which implies claim (i) immediately. Now consider a $t \in \mathcal{T}$. Then there is a $b = b_t > 0$ such that $I(b) = (b + (c-\mu)t)^2/(2v(t))$. We thus obtain claim (ii). Now consider claim (iii). We have to prove that for all $t > 0$ there is a $b > 0$ such that $t = t_b$. Evidently, t_b solves $2v(t)(c-\mu) = (b + (c-\mu)t)v'(t)$, or, equivalently, $b = b_t := (2v(t)/v'(t) - t)\,(c-\mu)$. Hence, it is sufficient if b_t grows from 0 to ∞ when t grows from 0 to ∞.

Note that even if condition (iii) does not apply, as was found in some recent traces, see [7], $v(t)$ will be bounded from above by the infimum over b, due to (i). Remarkably, Theorem 4 gives, loosely speaking, that for Gaussian sources the buffer content distribution uniquely determines the variance function. This property is exploited in the following heuristic.

Approximation 5. *The following estimate of the function $V(t)$ (for $t > 0$) can be made using the buffer content distribution:*

$$V(t) \approx \inf_{B>0} \frac{(B + (C-M)t)^2}{-2\log \Pr(Q > B)}. \tag{6}$$

Hence, if we can estimate $\Pr(Q > B)$, then 'inversion formula' (6) can be used to retrieve the variance; notice that the infimum can be computed for any t, and consequently we get an approximation for the entire variance curve $V(\cdot)$ (of course up to some finite horizon). These ideas are exploited in the procedure described in the next section.

4 Demonstration of the Inversion Procedure

In this section we show how the theoretical results of the previous section can be used to estimate $V(\cdot)$. First, we propose an algorithm for estimating the (complementary) buffer content distribution (in the sequel abbreviated to BCD), such that, by applying Approximation 5, the variance curve $V(\cdot)$ can be estimated. In our demonstration, second, we specialize to the case of synthetic input, i.e., traffic generated according to some stochastic process; we choose fBm input, but we emphasize that the procedure could be followed for any other process. Finally, we compare, for fBm, our estimation for $V(\cdot)$ with the actual variance curve, yielding a first impression of the accuracy of our approach (a more detailed numerical evaluation follows in Sect. 5 and 6).

The inversion procedure consists of two steps: (1) determining the BCD, and (2) 'inverting' the BCD to the variance curve $V(\cdot)$ by applying Approximation 5. We propose the following algorithm:

Algorithm 6. *Inversion approach.*

1. *Collect 'snapshots' of the buffer contents: $q_1, \ldots, q_N$; here q_i denotes the buffer content as measured at time $\tau_0 + i\tau$, for some $\tau > 0$. Estimate the* BCD *by the empirical distribution function of the q_i, i.e., estimate* $\Pr(Q > B)$ *by* $\phi(B) = \#\{i : q_i > B\}/N$.
2. *Estimate $V(t)$, for any $t \geq 0$, by* $\inf_{B>0}(B + (C-M)t)^2/(-2\log\phi(B))$.

In the above algorithm, snapshots of the buffer content are taken at a constant frequency. To get an accurate estimate of the BCD, both τ and N should be chosen sufficiently

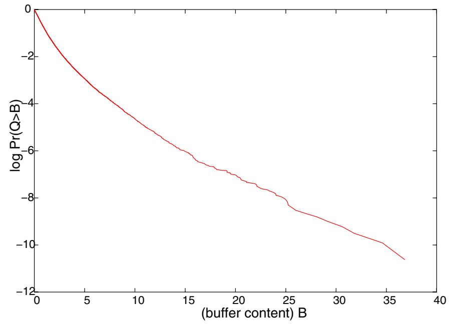


Fig. 1. Sample BCD

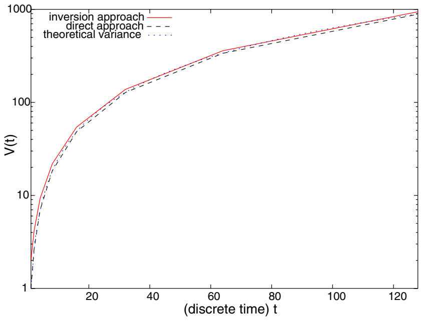


Fig. 2. Sample variance curves

large. We come back to this issue in Sect. 5. Notice that we chose a fixed polling frequency (i.e., τ^{-1}) in our algorithm, but this is not strictly necessary; the BCD-estimation procedure obviously still works when the polling epochs are not equally spaced.

In the remainder of this section we demonstrate the inversion approach of Algorithm 6 through a simulation with synthetic (fBm) input. The simulation of the queue fed by fBm yields an estimate for the BCD; this estimated BCD is inverted to obtain the estimated variance curve, which is compared with the actual variance curve.

Simulation procedure. Concentrating on slotted time, we generate traffic according to some stochastic process. In this example we focus on the case of fBm input, but it is stressed that the procedure could be followed for any other stochastic process. The traffic stream is fed into a simulated queue with link rate C. The buffering dynamics are simulated as follows: (1) using a fBm simulator [10], fBm is generated with a specific Hurst parameter $H \in (0,1)$, yielding a list of 'offered traffic per time slot'; (2) for every slot, the amount of offered traffic is added to, and an amount equal to C is drained from the queue (while assuring the queue's content is non-negative); (3) every τ slots, the queue's content is observed, yielding N snapshots that are then used to estimate $\Pr(Q > B)$ (cf. Algorithm 6). In this demonstration of the inversion procedure, we generate an fBm traffic trace with Hurst parameter $H = 0.7$ and length 2^{24} slots. The link capacity C is set to 0.8, and we take snapshots of the buffer content every $\tau = 128$ slots.

Estimating the variance curve. We now discuss the output of the inversion procedure for our simulated example with fBm traffic. First we estimate the BCD; a plot is given in Fig. 1. For presentation purposes, we plot the (natural) logarithm of the BCD, i.e., $\log \Pr(Q > B)$. The BCD in Fig. 1 is 'less smooth' for larger values of B. This is due to the fact that large buffer levels are rarely exceeded, leading to less accurate estimates. Second, we estimate the variance $V(t)$ for t equal to the powers of 2 ranging from 2^0 to 2^7, using the BCD, i.e., by using Algorithm 6. The resulting variance curve is shown in Fig. 2 ('inversion approach'). The minimization (over B) was done by straightforward numerical techniques. To get an impression of the accuracy of the inversion approach, we have also plotted in Fig. 2 the variance curve as can be estimated directly from the synthetic traffic trace (i.e., the 'direct approach' introduced in Sect. 3), as well as the real variance function for fBm traffic, i.e., $V(t) = t^{2H}$. Figure 2 shows that the three variance curves are remarkably close to each other. This confirms that the inversion approach is an accurate way to estimate the burstiness.

5 Error Analysis of the Inversion Procedure

In the previous section the inversion approach was demonstrated. It was shown to perform well for fBm with $H = 0.7$, under a specific choice of N and τ. Evidently, the key question is whether the procedure still works under other circumstances. To this end, we first identify the three possible sources of errors:

- The inversion approach is based on the *approximation* (2).
- $\Pr(Q > B)$ is *estimated*; there could still be an estimation error involved. In particular, we wonder what the impact of the choice of N and τ is.
- The procedure *assumes* perfectly Gaussian traffic, although real network traffic may not be (accurately described by) Gaussian.

We will now quantitatively investigate the impact of each of these errors on our 'indirect approach'. These investigations are performed through simulation as outlined in Sect. 4.

Approximation of the buffer content distribution. In Equation (2) an approximation of the BCD is given. As the inversion approach is based on this approximation, evidently, errors in (2) might induce errors in the inversion.

We first determine the infimum in the right-hand side of (2), which we consider as a function of B. In line with the previous section, we choose fBm input: $M = 0$ and $V(t) = t^{2H}$. Straightforward calculations now reveal that we can rewrite (2), viz.:

$$\log \Pr(Q > B) \approx -1/2 \cdot (B/(1-H))^{2-2H} \, (C/H)^{2H} \quad .$$

We verify how accurate the approximation is, for two values of H: the pure Brownian case $H = 0.5$, and a case with long-range dependence $H = 0.7$ (in line with earlier measurement studies of network traffic). Several runs of fBm traffic are generated (with different random seeds), 2^{24} slots of traffic per run. We then simulate the buffer dynamics. For $H = 0.5$ we choose link rate $C = 0.2$, for $H = 0.7$ we choose $C = 0.8$; these choices C are such that the queue is non-empty sufficiently often (in order to obtain a reliable estimate of the BCD). Figures 3 and 4 show for the various runs the approximation of the BCD, as well as their theoretical counterpart. It can be seen that, in particular for small B the empirically determined BCD almost perfectly fits the theoretical approximation.

Estimation of the buffer content distribution. As we estimate the BCD on the basis of snapshots of the buffer content, there will be some error involved. The impact of this error is the subject of this subsection. It could be expected that the larger N (more observations) and τ (less correlation between the observations), the better the estimate.

First, we investigate the impact of N. The simulator is run as in previous cases (with $H = 0.7$), with the difference that we only use the first $x\%$ of the snapshots samples to determine $\Pr(Q > B)$. Figure 5 shows the estimation of the buffer content distribution, for various x ranging from 0.1 to 100. The figure shows that, in particular for relatively small B, a relatively small number of observations suffices to get an accurate estimate of the BCD.

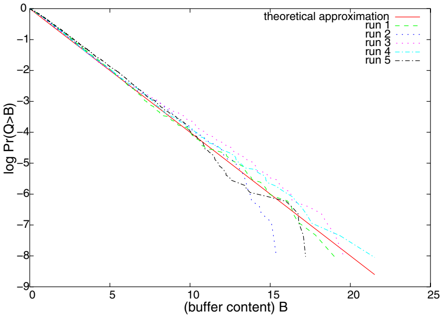


Fig. 3. $\Pr(Q > B)$ and theoretical approximation ($H = 0.5$)

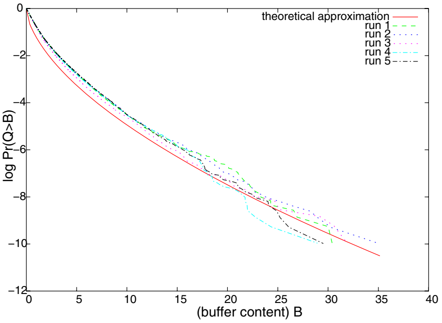


Fig. 4. $\Pr(Q > B)$ and theoretical approximation ($H = 0.7$)

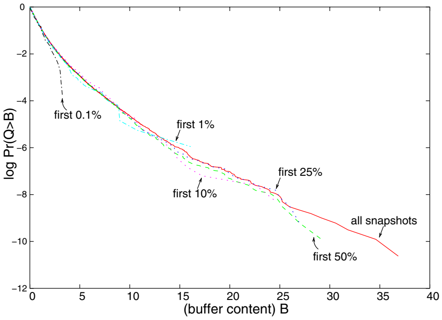


Fig. 5. Comparing $\Pr(Q > B)$ for various trace lengths

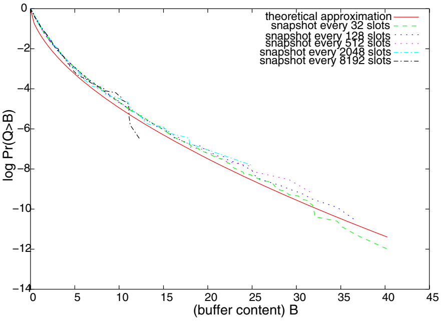


Fig. 6. Comparing $\Pr(Q > B)$ for various polling intervals

Second, we investigate the impact of the interval length between two consecutive snapshots τ. Figure 6 shows the determined BCD for τ ranging from observing every 32 to every 8192 slots. It can be seen that, particularly for small B the fit is quite good, even when the buffer content is polled only relatively rarely.

The impact of the Gaussianity assumption. Approximation (2) explicitly assumes that the traffic process involved is Gaussian. Various measurement studies find that real network traffic on the Internet is (accurately described by) Gaussian, see, e.g., [3]; others claim, however, that particularly on small time-scales, traffic may not be Gaussian [2]. Therefore we now investigate how sensitive our 'inversion approach' is with respect to Gaussianity of the input traffic.

We study the impact of non-Gaussianity by mixing, for every slot, a fraction α of the generated fBm traffic with a fraction $1 - \alpha$ traffic from an alternative (non-Gaussian) stream, before the mixture is fed to the (virtual) queue. Note that the variance of the mixture is $V(t) = \alpha^2 V_{[\text{fBm}]}(t) + (1-\alpha)^2 V_{[\text{alt}]}(t)$. We vary α from 1 to 0, to assess the impact of the non-Gaussianity.

The alternative input model that we choose here is an M/G/∞ input model, inspired by, e.g., [4, 8]. In the M/G/∞ input model, jobs arrive according to a Poisson(λ) process. The job durations are i.i.d., and during their duration each job generates traffic at a

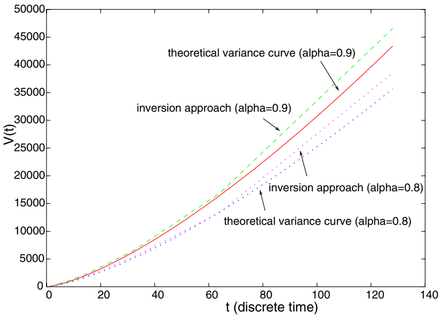


Fig. 7. Variance curves for Gaussian/non-Gaussian traffic mixtures, $\alpha = \{0.8, 0.9\}$

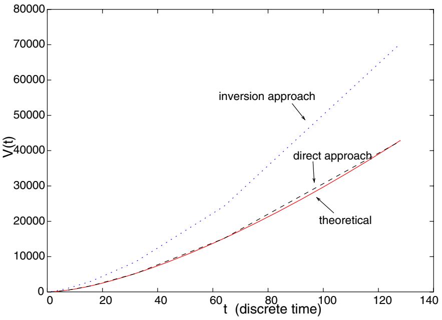


Fig. 8. Variance curves for Gaussian/non-Gaussian traffic mixture, $\alpha = 0$

constant rate r. In line with measurements studies, we choose Pareto(β) jobs. As the objective is to assess the impact of varying the parameter α, we have chosen to select the parameters of the M/G/∞ model such that the fBm and M/G/∞ traffic streams are 'compatible', in that their mean and variance $V(\cdot)$ are similar, which is achieved as follows.

The means of both traffic stream are made compatible by adding a drift to the fBm inputs equal to the mean of the M/G/∞ traffic stream, i.e., $\lambda r/(\beta - 1)$. Here λ and r can be chosen arbitrarily. The Gaussianity of the fBm input is not affected by the addition of such a drift.

To make the variances of both traffic streams compatible, we make use of a derivation in earlier work of the exact variance function $V(t)_{[\mathrm{alt}]}$, see [11]. It is not possible to achieve the desired 'compatibility' of the variance on all time scales. As long-range dependence is mainly a property of long time-scales, we choose to focus on these. For larger time scales, the variance of the M/G/∞ traffic stream roughly looks like $V_{[\mathrm{alt}]}(t) \approx 2r^2\lambda t^{3-\beta}/((3-\beta)(2-\beta)(\beta-1))$, assuming $\beta \in (1,2)$. We can now estimate the remaining parameters and compute the variance of the traffic mixture.

The next step is to run, for different values of α, the simulation. We then determine the (theoretical) variance curve of the traffic mixture, and compare it to the variance curve found through our 'indirect approach'. In Fig. 7 we focus on the 'nearly-Gaussian' cases $\alpha = 0.8$ and $\alpha = 0.9$, which are plotted together with their theoretical counterparts. The figure shows that the presence of non-Gaussian traffic has some, but no crucial impact on our inversion procedure. We also consider the (extreme) case of $\alpha = 0$, i.e., no Gaussian traffic at all, to see if our inversion procedure still works. In Fig. 8 the various variance curves are shown: the theoretical curve, the curve based on the 'direct approach', as well as the curve based on the inversion approach. Although not a perfect fit, the curves look similar and still relatively close to each other (but, of course, the fit is worse than for $\alpha = 0.8$ and 0.9). Note that the non-Gaussian traffic may 'have some Gaussian characteristics' if there is a large degree of aggregation, by virtue of central-limit type of arguments, which may explain that the fit is still reasonable.

We conclude that our simulation experiments show the 'robustness' of the inversion procedure. Despite the approximations involved, with a relatively low measurement effort, the variance curve is estimated accurately, even for traffic that is not perfectly Gaussian. Given the evident advantages of the inversion approach over the 'direct approach' (minimal measurement effort required, retrieval of the entire variance curve $V(\cdot)$, etc. – see the discussion in Sect. 3), the former method is to be preferred. In the next section we verify whether this conclusion also holds for real (i.e., not artificially generated) network traffic.

6 Empirical Validation of the Inversion Procedure

Whereas the previous sections studied the performance of our inversion approach by executing simulation experiments with synthetic traffic, in this section we use traces of real network traffic. We evaluate the inversion approach by comparing with the 'direct approach'.

Measurement and simulation setup. The traces used here are collected at the so-called (Ethernet) uplinks of five distinct networks (i.e., links that connect these networks to their Internet service providers). These networks resemble various scenarios, such as different types of users (e.g. students, 'normal consumers', and web-servers), and different access techniques (ranging from 512 kbps ADSL to 100 Mbps Ethernet), in order to test our 'indirect approach' under different circumstances. For each network, we have hooked up an off-the-shelf PC to a router/switch that copies all traffic from/to the uplink to the measurement PC. Using the standard tcpdump software, all packet headers are captured, time-stamped [9], and subsequently made anonymous through the tcpdpriv tool to protect the users' privacy. In this way we have obtained over 400 traces in total, each of them containing 15 minutes of traffic.

The collected packet traces are used in two ways: (1) to estimate the variance curve through the 'direct approach'; and (2) to 'replay' the traffic through a virtual queue and link, analogous to the simulation procedure described in Sect. 4, and then applying the 'indirect approach' to estimate the variance, cf. Algorithm 6.

Empirical validation. Because of paper length restrictions, we cannot discuss here the empirical validation for all traces, nor all networks that we have measured. Therefore we resort to presenting only the validation of two of the hundreds of traces; empirical validation of the other traces has shown similar results. The first example is based on a trace captured from the 1 Gbps uplink of an ADSL access network that has about 800 subscribers. The second example is about a web-server farm, with approximately 150 servers; access is 100 Mbps Ethernet, and the uplink is 30 Mbps.

We have set the 'sampling interval' τ (used to estimate the BCD) to 1 second, to ensure that a considerable number of snapshots (900) can be taken. A substantial fraction of these snapshots, also depending on the value of the output link's capacity, do not provide any information as the buffer turns out to be empty at the time the snapshot is taken. We choose the smallest interval length for which we compare the variance estimated through our inversion approach with the actual variance found through the 'direct approach', to be 10 ms (which is, in other words, 100 times as small as the interval we poll the buffer occupancy).

Figures 9 and 10 show the variance curves for the first and second example, respectively. Clearly, the graphs demonstrate that the inversion approach is capable of adequately estimating $V(\cdot)$. Hence even for real network traces, of which for instance the Gaussian character is far from evident, our coarse-grained buffer polling approach suffices to estimate $V(\cdot)$. We recall that this implies that this eliminates the need for doing

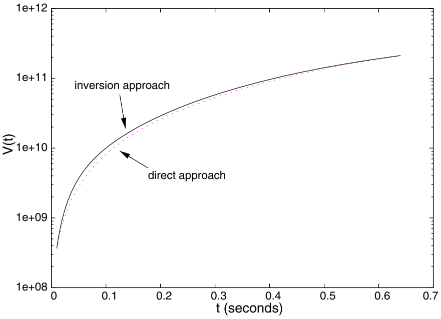


Fig. 9. Variance curves, ADSL access net

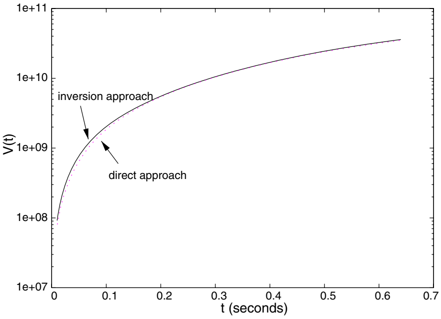


Fig. 10. Variance curves, web-server farm

detailed, small-time-scale, traffic measurements. In particular, our results indicate that we can estimate the variance on the time-scale 10 ms, just by polling the buffer content at a low frequency, without ever doing a traffic measurement on the 10 ms time scale.

7 Discussion

After the numerical tests of the previous sections, the next issue concerns the feasibility of implementing the inversion approach in operational environments.

The above experiments (both with artificial traffic and real traces) have been done off-line, in that we have used scripts that 'parse' the synthetic and real traffic, mimicking the buffer content dynamics. An interesting question is whether this approach is feasible in run-time, in operational environments. From the proposed procedure, we can derive three functional requirements for an implementation of the inversion procedure.

First, a notion of the amount of data in a buffer. This has already been addressed in real Internet routers: Random Early Detection (RED) [12] queuing algorithms, which are widely implemented in modern routers, also keep track of the amount of queued data. Second, a way to poll the buffer occupancy; SNMP may be used for this (in case the entire procedure is not run on the router itself). Third, software/hardware to determine the BCD, and then determine the resulting estimate of $V(\cdot)$. This has also already been addressed – see our results in this paper.

The above aspects lead us to believe that there are no fundamental or conceptual problems preventing the actual application of our approach in practice.

8 Concluding Remarks

We have presented a novel method to determine the burstiness of network traffic; here burstiness is in terms of the variance $V(T)$ of the traffic generated in an arbitrary window of length T. Our approach estimates the entire variance curve $V(\cdot)$ (of course up to some horizon), also on small time-scales, *without performing detailed traffic measurements*. Instead, the buffer content is polled (at some coarse-grained frequency) to obtain an estimate of the buffer content distribution. Then this distribution is 'inverted' to find the variance curve of the traffic rates, which gives the 'burstiness' $V(T)$ of the network traffic *at any time scale T* (up to some horizon). Knowledge of the variance curve can immediately be used in provisioning formulae. We have presented the mathematical foundations under this inversion method, and have investigated its accuracy by performing a thorough analysis of the possible sources of error. Although there exists an assumption on Gaussianity of the traffic, the error analysis has shown that even when the traffic is not perfectly Gaussian, our inversion method gives good results.

Furthermore, we have validated our approach in various real-life settings. This has shown that our inversion method provides remarkably accurate estimates of the traffic's burstiness. In particular, we have observed that our approach yields reliable estimates of the variance for very small time-scales. A seemingly counterintuitive but representative example: by sampling from the buffer occupancy every second we have found an accurate estimate of the variance on the time-scale of 10 ms.

In future work we intend to investigate the impact of the precise choices of the queue's link rate C on our inversion approach. In particular it would be interesting to optimize C such that the fit is optimal (with respect to some optimality criterion). We also intend to test our approach in even more real-life settings, and extend the concept to a network setting.

References

1. van de Meent, R., Pras, A., Mandjes, M., van den Berg, H., Roijers, F., Venemans, P., Nieuwenhuis, L.: Burstiness predictions based on rough network traffic measurements. In: Proc. 19th World Telecommunications Congress (2004)
2. Kilpi, J., Norros, I.: Testing the gaussian approximation of aggregate traffic. In: Proc. Internet Measurement Workshop (2002)
3. Leland, W., Taqqu, M., Willinger, W., Wilson, D.: On the self-similar nature of Ethernet traffic (extended version). IEEE/ACM Trans. on Networking **1** (1994) 1–15
4. Addie, R., Mannersalo, P., Norros, I.: Most probable paths and performance formulae for buffers with gaussian input traffic. European Trans. Telecommunications **13** (2002) 183–196
5. Fraleigh, C., Tobagi, F., Diot, C.: Provisioning IP Backbone Networks to Support Latency Sensitive Traffic. In: Proc. IEEE Infocom, (2003)
6. Beran, J.: Statistics for Long-Memory Processes. Chapman & Hall/CRC (1994)
7. Zhang, Z., Ribeiro, V.J., Moon, S., Diot, C.: Small-Time Scaling Behaviors of Internet Backbone Traffic: An Empirical Study. In: Proc. IEEE Infocom (2003)
8. Fredj, S.B., Bonald, T., Proutiere, A., Régnié, G., Roberts, J.W.: Statistical Bandwidth Sharing: A Study of Congestion at Flow Level. In: ACM SIGCOMM CCR **4** (2001) 111-122
9. van de Meent, R., Pras, A., Mandjes, M., van den Berg, H., Nieuwenhuis, L.: Traffic Measurements for Link Dimensioning: A Case Study. Proc. of the 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM2003) (2003) 106–117
10. Dieker, T.: (Fractional Brownian motion simulator) `http://homepages.cwi.nl/~ton/fbm/index.html`.
11. Mandjes, M., Saniee, I., Stolyar, A.: Load characterization, overload prediction and load anomaly detection for voice over IP traffic. In: Proc. 38th Allerton Conference (2000)
12. Floyd, S., Jacobson, V.: Random Early Detection (RED) gateways for Congestion Avoidance. IEEE/ACM Trans. on Networking **1** (1993) 397–413

Modeling Available Bandwidth for an Efficient QoS Characterization of a Network Path*

Alexander Chobanyan[1], Matt W. Mutka[1], V.S. Mandrekar[2], and Ning Xi[3]

[1] Department of Computer Science and Engineering, Michigan State University
{chobany1, mutka}@cse.msu.edu

[2] Department of Statistics and Probability, Michigan State University
mandrekar@stt.msu.edu

[3] Department of Electrical and Computer Engineering, Michigan State University
xin@egr.msu.edu

Abstract. Estimating the reliability of an end-to-end network path is critically important for applications that support remote real-time task execution. Available bandwidth, which is defined as a minimum spare capacity of links constituting a network path, is an important QoS characteristic of the path. In this work we demonstrate a new approach to modelling available bandwidth behavior from a time-series analysis prospective. In particular, we introduce a notion of *crossing probability*–the probability that available bandwidth drops below the QoS critical threshold for the period of time required for a real-time task execution. We estimate "crossing probability" by an application of the $ARCH^2$ (AutoRegressive Conditional Heteroscedasticity) model to available bandwidth behavior. We estimate model coefficients β_0 and β_1 to quickly output "crossing probability" for arbitrary values of threshold and length of the real-time task. The model was evaluated on real bandwidth measurements across multiple network paths.

Index terms: Network measurements, Statistics, Stochastic processes.

1 Introduction

Over the past several years the importance of Internet-related technologies have rapidly increased. The growth of the Internet has not only a quantitative nature but also affects the type of information that may be transferred and consequently the methods for real-time interaction between communicating parties. Many rapidly emerging network applications may benefit not only by transferring traditional media-types, such as video and audio, but also by remotely manipulating, touching and feeling a remote object. One may imagine the influence of new media types when new progressive technologies emerge, such as tele-medicine that allows physician to remotely touch and feel a patient, tele-education that opens wider possibilities to distant learning about various objects

* This research is supported in part by NSF Grant No. 0334035.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 316–327, 2005.

via sensing, and tele-commerce that allows a customer to feel and manipulate an object before conducting a financial transaction.

Extending transferable media to new types, such as haptic and temperature data, give rise to many challenges such as channel synchronization, increased real-time requirements for tele-operated task execution, and others. In particular, the nature of real-time tasks related to new media types requires Quality-of-Service (QoS) guarantees to communicating parties over the potentially unreliable and uncontrolled Internet. Available bandwidth is an important QoS characteristic of the network path. It may be briefly defined as the maximum rate that a sender may put bits on a medium without affecting existing cross-traffic, or in other words, as a residual capacity of the network path. In this paper we present a statistical model for a time-series that represents the available end-to-end bandwidth. The proposed model provides an important *probabilistic* QoS channel specification in terms of coefficients that allow the rapid computation of *crossing probability*–the probability that the available bandwidth becomes less than an arbitrary pre-defined critical value over an arbitrary pre-defined time-frame. The aforementioned coefficients are evaluated based on observed available bandwidth behavior over a reasonable period of time and are updated at run-time reflecting the changing dynamics of a network environment. The critical threshold and the time-frame should be defined by the QoS specifications of a particular real-time task.

The available bandwidth time-series is directly related to the cross-traffic that is present in the path. It has been shown [1] that network traffic has self-similar properties, which imply a non-summable autocorrelation function and a long-range dependence between observations. Therefore observations of available bandwidth cannot be considered as independent, which makes statistical inference more challenging. All previous work [2] concentrated on obtaining the value of available bandwidth averaged over a reasonably small time-window without an attempt to provide a broader probabilistic picture that characterizes the network path. Figure 1 shows an example of how the available bandwidth between two nodes may behave over time. As depicted in the plot, the available bandwidth may instantaneously drop far below its mean value, thus causing degradation of service, such as distorted video or audio. Despite the fact that such "spikes" may last only a few seconds, they may be fatal for many real-time remotely-operated tasks. As illustrated in figure 1, the value of the available bandwidth averaged over an arbitrary large (or small) time-window cannot characterize the network path well in terms of its bandwidth QoS properties.

The key difference between our approach and recent work is that we apply random time-series statistical techniques to available bandwidth behavior. We create a probabilistic framework for network path QoS characterization. A number of various statistical models that consider dependence between observations may describe instantaneous spikes and reasonably predict the "crossing probability" based on the chosen model. Many successful time-series analysis techniques have been applied to similar applications in econometrics and natural sciences. We note, in particular, that the ARCH model used in econometrics is able to describe spikes and possesses self-similar properties, which is important for making inferences about the "crossing" probability.

The notion of "crossing probability" on its own gives rise to conceptually novel approach to available bandwidth estimation. Real time applications may benefit from this approach by reallocating part of their QoS-sensitive channels over different network paths when the *crossing probability* exceeds an acceptable threshold. Channel reallocation may happen before something fatal occurs for a real-time task but after the probability for this fatal event happens to exceed a reasonable threshold.

2 Background

A network path may be viewed as a set of store-and-forward links $L_i, i = 1, 2, ..., k$. The capacity of the i-th link C_i may be defined as the maximum bit rate that the sender may transfer information through this link when no other traffic is present. Assume that the link is partially used by a separate communication. Let $T_i(t)$ be defined as the number of bits transferred through this link by time t. The spare capacity of link i, or in other words, its available bandwidth $A_i(t)$ at time t may be defined as $A_i(t) \equiv C_i - \frac{dT_i}{dt}(t)$. The bandwidth $A(t)$ available at time t of a path may be defined as $A(t) \equiv \min_{i=1,...,k} A_i(t)$. From this definition it follows that the available bandwidth cannot be measured precisely and therefore needs to be approximated by the mean available bandwidth at time t for the interval τ: $A_\tau(t) = \min_{i=1,...,l}\{C_i - \frac{T_i(t+\tau)-T_i(t)}{\tau}\}$. With smaller τ, one may achieve better flexibility in describing the available bandwidth behavior. In general, the approximation of available bandwidth is QoS relevant if τ is less than the time that receiver needs to empty its buffer. Fluctuations of the available bandwidth should not cause degradation of quality at the receiver as long as those fluctuations do not affect the number of bits that the receiver needs to keep its buffer non-empty.

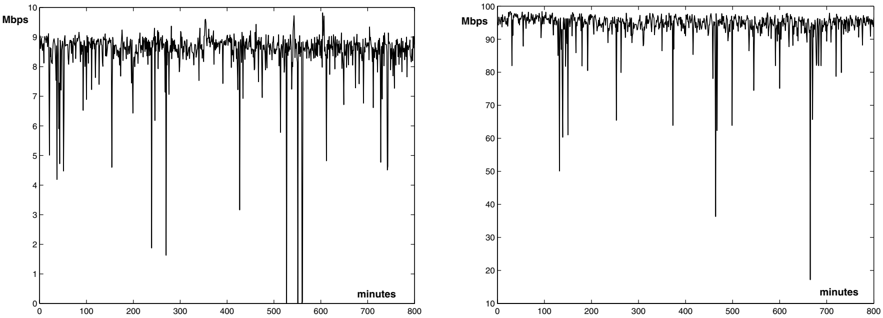


Fig. 1. Estimated available end-to-end bandwidth (Mbps) across different network paths. The time-series are sampled at a rate of one observation per minute

Significant work conducted regarding mean available bandwidth estimation is well reflected by Strauss, et al. [3] and Hu, et al. [4]. Probe gap models used in Spruce [3] and Delphi [5] measure delay between probe packets and make further inference about available bandwidth based on that delay. As it was shown by Prasad et al. [6] context switches and enabled "interrupt coalescence" feature at receiver's side may distort measurements of a gap between packet pairs and therefore significantly influence probe-gap based available bandwidth measurements.

The probe rate model used in Pathload [2] use sequences or "trains" of packets sent at different rates. By detecting a trend in packet transmission times sent at a given rate Pathload decides whether the specified rate is higher or lower than the available bandwidth. Then the available bandwidth can be found by a simple binary search. Pathload is believed to be more intrusive than tools using the probe gap model. To our knowledge Pathload is, however, the only tool that considers effect of context switches and interrupt coalescence at the receiver side and is therefore the most stable and accurate among available bandwidth measurement tools to date.

From the definition of available bandwidth, it follows that it is closely related to cross-traffic in the network path. Cross-traffic, defined as $\frac{dT}{dt}(t)$, represents a random time-series that has been extensively analyzed by both statisticians and network engineers for its statistical properties. If the most tight link remains unchanged throughout measurements (which is usually the case), then the available bandwidth time-series may be evaluated by subtracting the cross-traffic from a tight link's unchanged capacity. It follows that in the aforementioned case the available bandwidth and cross-traffic are exactly the same time-series in terms of their statistical characteristics. Therefore we believe that the model chosen for available bandwidth characterization should incorporate at least basic properties of models that are presently used for Internet traffic modelling. Below we provide some background of such models.

All parametric traffic models assume weak stationarity of the traffic time-series, which means that the correlation between time-series observations remains the same under any shift in the time domain. Leland, et al. [1] showed in 1994 that network traffic possesses an important characteristic called self-similarity. Most important implication of self-similarity is "slowly" (slower than $1/h$) decreasing correlation between observations located at lag h apart implying long-range dependence.

Many recent works in internet traffic modelling [7, 8, 9] confirmed that self-similarity is one of key-importance properties of a present Internet traffic. Researchers tried to build models that visually look similar to observed time-series and have an autocorrelation structure similar to what has been observed. Model parameter estimation based on observed data should remain unchanged or vary slowly over time. Furthermore, the model should be relatively simple and amenable for the further statistical analysis. We follow the same strategy to unify recent work in Internet traffic analysis, available bandwidth estimation tools, and time-series studies with respect to the analysis of QoS characteristics of a network path.

3 Data Collection and Preprocessing

For this work we made continuous measurements of available bandwidth between a number of machines located in Michigan State, Berkeley, MIT, University of Sydney, and Uupsala University (Sweden), thus covering a broad range of geographical locations. All machines chosen for the experiment had very low processor utilization factor since the Pathload measurements are very sensitive to processor load of a machine responsible for receiving trains of measurement packets (context switch / interrupt coalescence effect). Figure 1 depicts the available bandwidth behavior for the period of 800 minutes of a few network paths. The distance between nodes varies from 4 to 20 hops. In each example in figure 1 we observe a small trend that is 2 Mbps around the mean value. Each example has bursts reflecting the heavy-tailed self-similar nature of Internet traffic-related series. For further fitting of stationary self-similar models we preprocess time-series of interest by subtracting the trend. For all examples considered in figure 1, trend-subtraction changes each observation for less than 2 Mbps, which is negligible with respect to spikes that are the primary cause for instantaneous unexpected degradation of service. Predicting the probability of such a spike to occur over a certain a QoS sensitive time-slice is our primary goal. For the aforementioned purpose we can neglect 2-3 Mbps trends and further consider the "adjusted" time-series. Figure 2 shows the original time-series representing available bandwidth between nodes located in MSU and Berkeley along with the corresponding "adjusted" time-series. Figure 2 shows that trend-removal, inversion and "lining-up" to zero level do not significantly change the behavior pattern of the original time-series of interest. For trend-removal, we used the Spencer 15-point moving average filter described by Brockwell, et al. [10], which skips polynomials of degree three and smoothes everything that has a larger degree.

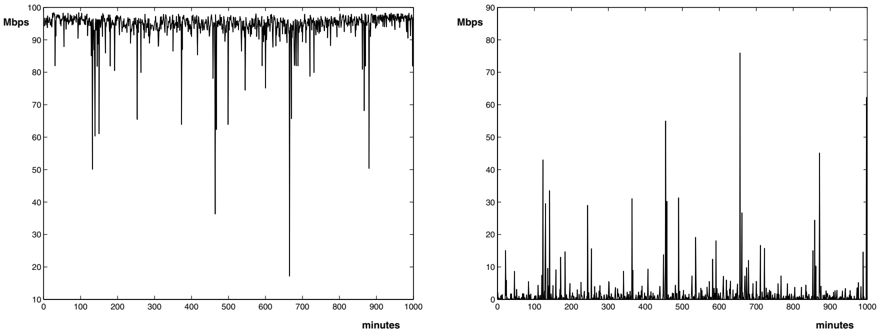


Fig. 2. Estimated available end-to-end bandwidth (Mbps). Trend and inversion effect

It is also undesirable to have instantaneous spikes affecting trend estimation. Therefore a "trimming" procedure is applied before passing data through a Spencer filter. The whole procedure may be done at run-time. The size of the window that slides across the time-series observations is 19. We delete two maximal and two minimal observations, and pass the remaining 15 observations to a Spencer filter that outputs the estimated trend. Then trend is subtracted from the original time-series, the result is inverted, lined-up to zero, and thus finally the "adjusted" time series is obtained.

The following discussion will be related to analysis of the "adjusted" time-series. Note that in most cases the "adjustment" procedure does not significantly distort data. Even if it does, we may detect a bad on-going trend, report the potential QoS problem to the communicating parties, and still learn at the same time about the pattern of spikes.

To our knowledge rare cases fall beyond the scope of this work, such as when there exists long-lasting constantly present trend exceeding 5 Mbps or when routers on the network path frequently change routes for packets thus making available bandwidth oscillating around multiple different levels.

4 Modeling Available Bandwidth

4.1 "Crossing Probability"

Assume that an operator is about to decide whether to perform a real-time QoS critical task over a potentially unreliable network path. The most important question may be stated as follows. What is the probability of experiencing QoS critical degradation of service during the time required for a given real-time task execution? An example is the probability that a picture transferred to the operator from a remotely-controlled walking robot distorts to a point that the operator is not able to understand where the robot is located with respect to an approaching wall.

Let us slightly formalize the problem. The natural assumption is that the operator should be aware of the real-time task duration, minimal allowed transmission rate and the receiver's buffer size. These three metrics may be considered as basic QoS characteristics of a given task. QoS requirements are violated if the available bandwidth averaged over time $\tau = \frac{receiver's\ buffer\ size}{minimal\ allowed\ trasmission\ rate}$ drops below the threshold of minimal allowed transmission rate during the task's execution time. Note that the task's execution time T corresponds to a sequence of observations of mean available bandwidth time-series of a length $N = \frac{T}{\tau}$. Now rephrase the main QoS question as follows: What is the probability that the minimal value over the next N consecutive observations of available bandwidth time-series drops below a given threshold where both threshold and N are specified by real-time task constraints? For the "adjusted" and inverted mean available bandwidth time-series, the aforementioned probability may be re-defined as *the probability that the maximum over the next consecutive N observations of the time-series crosses the pre-defined QoS critical threshold.* We believe that finding this probability should be an ultimate goal and statistically correct problem statement for the QoS characteristics of a network path. We here-and-after refer to this probability as to a "crossing probability". Even finding a reasonable upper bounds for such probability is a worthwhile contribution to the QoS characterization of a network path. Below we build a model that describes the observed available bandwidth data. The model is self-similar and amenable to analysis with respect to the aforementioned "crossing probability."

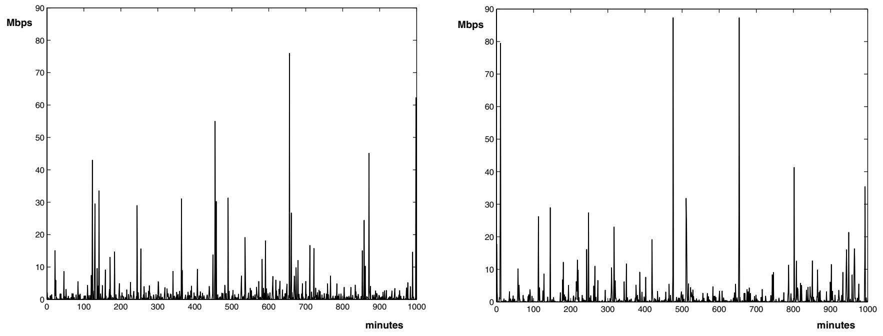
4.2 ARCH Model

We propose a parametric model that allows efficient computation of the "crossing probability" for any given number of future observations N and the threshold M based on the estimated model parameters. We used the "$ARCH^2(1)$" model [11] for description of the mean available bandwidth data. Therefore we first provide background on ARCH models and describe reasons why this model has been chosen as a candidate for the mean available bandwidth behavior description.

The ARCH(p) (Auto-Regressive Conditional Heteroscedasticity) time-series model was introduced by Engle [11] for applications of financial events. The ARCH(p) model may be defined as an *autocorrelated* Gaussian noise with dependency between observations described as follows. $X_t = \sqrt{\beta_0 + \sum_{i=1}^{p} \beta_i X_{t-i}^2} Z_t$ where Z_t are independent observations following a standard normal distribution. The simplest partial case is the $ARCH(1)$ time-series defined respectively as $X_t = \sqrt{\beta_0 + \beta_1 X_{t-1}^2} Z_t$. Finally, the $ARCH^2(1)$ time-series will be defined as $Y_t = (\beta_0 + \beta_1 Y_{t-1}) Z_t^2$. From the definition of $ARCH^2(1)$, it follows that the model is completely defined by a pair of coefficients $[\beta_0, \beta_1]$. Both $ARCH(1)$ and $ARCH^2(1)$ models were extensively applied to finance-related time-series that at least visually have the similarity to the "adjusted" mean available bandwidth time-series pattern. Before going any further, we provide a picture with "adjusted" mean available bandwidth on the left and a simulated $ARCH^2(1)$ time-series on the right. As shown in figure 3, the patterns of both time-series are the same. Another encouraging factor is that it has been shown by Embrechts, et al. [12], that the $ARCH^2(1)$ time-series possesses self-similar characteristics.

In order to apply a parametric model to "crossing probability" estimation, one needs to ensure that the model parameters may be efficiently estimated at run-time. The model needs to be mathematically analyzed in order to find an efficient method to compute the "crossing probability" based on the estimated model parameters. In addition, one needs to build confidence intervals around the estimated probability in order to be aware of the model's accuracy. Then the model needs to be empirically verified. We therefore provide below our work on unification of the results in $ARCH^2(1)$ parameters estimation and "crossing probability" computation together with the scheme of confidence interval construction.

The basic work in ARCH parameter estimation has been conducted by Weiss [13]. Weiss [13] proposes to estimate ARCH parameters by maximizing Quasi Log Likelihood Function evaluated as: $L(\beta_0, \beta_1) = \sum_{i=1}^{n-1} [\log(\beta_0 + \beta_1 X_i^2) + \frac{X_{i+1}^2}{\beta_0 + \beta_1 X_i^2}]$ Here n is the number of observations in the window that is used for parameters estimation. The parameter n is very important. We discuss its proper adjustment separately. The advantage of the QMLE (Quasi Maximum Likelihood) estimator of $[\beta_0, \beta_1]$ is



(a) Measured available end-to-end bandwidth (Mbps)

(b) Simulated $ARCH^2(1)$ observations with $\beta_0 = \beta_1 = 0.25$

Fig. 3. Simulated versus observed series

that it is asymptotically unbiased. Also, it is known [14] that $[\hat{\beta}_0, \hat{\beta}_1]$ is normally distributed with mean $[\beta_{0true}, \beta_{1true}]$ and the covariance matrix is called Fisher Information. The normality of estimators gives background for computing the confidence interval of any level around the estimated values of model parameters.

Since estimators of $[\beta_0, \beta_1]$ depend on past observations and therefore have a random nature, we also provide a way for rapid run-time computation of the confidence region of any level (by default 95% CR). The 95% confidence region for $[\beta_0, \beta_1]$ is defined as a random region in R^2 that contains the point $[\beta_0, \beta_1]$ with probability ≥ 0.95. This region for the QMLE estimator has an elliptical form defined by the equation obtained by Hotelling [15] $(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})' \hat{\Sigma}^{-1} (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})(n-2)/2 = F_{2,n-2}(0.95)$ where $F_{2,n-2}(0.95)$ is the 95% quantile of the F-distribution with 2 and $n-2$ degrees of freedom, and $\hat{\Sigma}$ is the estimated Fisher information. In order to compute a confidence region quickly at run-time, we tabulated $\hat{\Sigma}$ for a fixed $n = 720$ and a grid of β_0 and β_1. Fisher information was calculated by simulating 1000 independent $ARCH^2(1)$ time-series for each fixed pair of β_0, β_1 and finding the difference between estimated and true values of the aforementioned vector. β_0 and β_1 then were varied within the region $(0.5, 3.5)$ and $(0, 0.6)$, respectively. This region covers all potentially acceptable values that β_0 and β_1 might take while describing the available bandwidth behavior within the range of 200 Mbps. It is known that the estimated covariance matrix is proportional to n^{-1}. Therefore, when having available tables for $n = 720$, one can recompute $\hat{\Sigma}$ for an arbitrary n in less than a second.

4.3 Results from Extremal Values Theory

Below we show mathematical results that allow computation of "crossing probability" for any specified threshold M, number of observations N, and estimated model parameters β_0, β_1. These results allow us to construct a 95% confidence interval for a "crossing probability" by considering the fact that the "crossing probability" is a monotonically increasing function of both β_0 and β_1 and that we find sufficiently many "crossing probability" points related to the 95% ellipse in the β_0, β_1 space. Embrechts, et al. [12] summarize recent work in extreme values by providing formulas for finding "crossing probability" for various time-series models. In particular, for $ARCH^2(1)$ model Embrechts, et al. have $\lim_{N \to \infty} P(N^{-\frac{1}{\kappa}} \max\{Y_1, ..., Y_N\} \leq M) = \exp^{-C_1 M^{-\kappa}}$ that allows us to rewrite this result for "crossing probability" P and large N as follows $P = 1 - \exp^{-C_1 M^{-\kappa} N}$ where κ is a unique root of the equation $\frac{(2\beta_1)^u}{\sqrt{\pi}} \Gamma(u + \frac{1}{2}) = 1$. Although Embrechts et al. [12] give an expression for C_1, it is more efficient to estimate it empirically off-line for different β_0 and β_1. For that purpose we simulated 800,000 observations for each pair of β_0 and β_1 coming from a grid with range $(0.5, 3.5)$ and $(0, 0.6)$, respectively. Then, based on the probability expression above, this tabulated result may be quickly generalized on-line to arbitrary N and M.

Calculating the estimated probability together with its confidence interval gives an opportunity to judge the size of the window n used for prediction of β_0 and β_1 coefficients. On one hand we want n to be as small as possible because we assume that model parameters remain unchanged for the time-period related to n. On the other hand, with larger n, accuracy of the prediction increases with corresponding shrinking of the 95% confidence region. In our experiments, for example, we put the following condition to the 95% confidence interval for the probability: the difference between the 95% upper bound and the predicted value should not exceed 0.2 for an arbitrary predicted probability value from the range [0;0.5]. Then for $N \in \{1, ..., 100\}$ and $M \in [60; 90]$ the lower boundary for n will be evaluated to 120. Note that dependence on N is natural: with increasing the period of time for which the prediction is made (duration of real-time task execution), the period of time used for prediction also needs to be correspondingly increased.

5 Experimental Evaluation

In this section we describe our method for testing the accuracy of our approach. The final target of modelling is prediction of a *probability*. The only way, therefore, to empirically verify the model is to apply the ergodic theorem, i.e., to have B groups with N observations in each and then count the number L of groups where the maximum over N observations within the group crosses a pre-defined threshold. Then the ratio L/B may be considered as an empirically computed value of a "crossing probability." We considered as our design accuracy the following criteria: within the frames of the model the difference between the predicted probability and its 95% upper confidence limit should not exceed 0.2 at any circumstances. Our simulations of $ARCH^2(1)$ series show that such "within-the-model" accuracy on a probability scale for $N \leq 90$ dictates the lower bound on the number of observations needed for model parameters estimation as $n = 120$. Higher accuracy leads to higher values of a lower bound for n.

In real-life, however, it is impossible to achieve such an ideal design of the experiment because of the following factors. First, one obtains B groups of measurements in a real scenario that belong to the same time-series (instead of having multiple independently generated series in a simulation scenario) and therefore are somehow dependent. Second, even if we assume that the model parameters remain the same over the estimation window of length n (that is reasonable if n correspond to 2 hours) it is too naive to assume that these parameters will remain the same over the whole time-period needed for collection of B groups of size N. Presently available bandwidth measurement tools allow consistently stable measurements at a rate of 1 sample/minute. Therefore, $B = 100$ reasonably separated groups of $N = 100$ observations in a group correspond to time needed for collection of 20,000 consecutive observations. With presently available sampling rate this time is evaluated to 20,000 minutes or 2 weeks of consistent measurements.

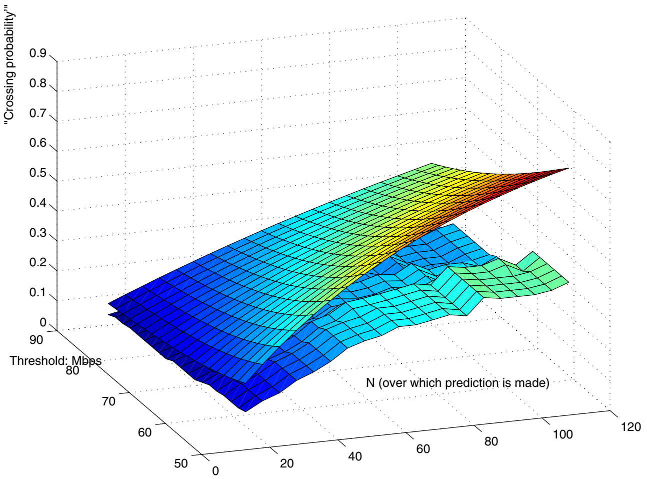


Fig. 4. Empirically computed "Crossing" probability within the frames of estimated 95% upper bound

The efficiency of the upper bound on the "crossing probability" may be, nevertheless, evaluated as follows. We considered a number of network paths that have visually unchanged over time their daily behavior pattern. Below we show as an example the analysis of one such path between nodes in MSU and Berkeley. We consider first 1440 observations (1 day) as our training-validation dataset. The rest 18,560 observations constitute testing dataset. Training observations were used for evaluation of the width of the strip where the estimated probability resides. The testing dataset was used for computation of the empirical probability L/B. We expected an average probability estimated with the testing data to reside in a 95% confidence strip defined by a first day training observations.

Figure 4 shows two surfaces where the upper surface represents the 95% upper bound estimated by the model and the lower surface represents the empirically computed probability. "Crossing probability" was computed for different values of N and M varying within the range 15-100 observations and 60-90 Mbps, respectively. History size n was taken as it was mentioned above equal to 120. As was expected, for all thresholds M and sizes N the empirically computed probability reside below the estimated bound. Figure 4 shows, in particular, that two surfaces are nearly parallel to each other, i.e., the model reflects scales of both N and M in reasonable agreement with what was empirically observed. Also figure 4 shows that the estimation becomes less accurate when N approaches to n, which is understandable.

We conclude that the model produced verifiable and an accurate prediction. We also note that such prediction is our suggested form of characterizing a present state of the network path with respect to its QoS properties.

6 Results Assessment and Future Work

We presented a "probabilistic" approach to a description of the available bandwidth behavior. We showed that the available bandwidth may be described by $ARCH^2(1)$ parametric model. We have studied and unified known results in $ARCH^2(1)$, tabulated relevant characteristics and provided an efficient method for computation of a "crossing probability". We tested the accuracy of our model on a channel with a relatively stable long-term behavior and proved that the approach has a range of applicability.

When describing available bandwidth from a time-series analysis prospective, it is important to have the time τ over which measurements are averaged as small as possible. Presently we are modifying and testing Pathload to increase its sampling rate to 10 samples/minute. With such a measurement tool, we may significantly decrease the total measurement time necessary to obtain the number of observations sufficient for the further statistical analysis. Recall that τ should be of the order of the receiver's buffer time in order to allow observations to be relevant with respect to a particular real-time task execution. Therefore, the real importance of the results provided in this work as well as the real range of their applicability may be extended after Pathload's proper modification.

Despite present challenges in available bandwidth measurement and determining the range of applicability of a particular parametric model, we see the main contribution of our approach in the correct "probabilistic" context of the problem statement. For real-time network applications, it is much more important to have at least tentative knowledge about the "crossing probability" rather than being aware of a single number representing the mean available bandwidth averaged over an inexactly defined time.

The notion of "crossing probability" opens research directions to predictive dynamic channel reallocation and resource planning for real-time network applications.

The aforementioned problems are subject of our future research. We, however, believe that this work is a significant step towards building a reliable QoS infrastructure over potentially unreliable media in order to facilitate the performance of many present real-time network applications.

References

1. W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, 1994.
2. M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *ACM SIGCOMM*, 2002.
3. J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *IMC*, 2003.
4. N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth techniques," in *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 2003.
5. V. Ribeiro, M. Coates, R. Riedi, S.Sarvotham, and R. Baraniuk, "Multifractal cross traffic estimation," in *ITC specialist seminar on IP traffic Measurement*, 2000.
6. R. Prasad, M. Jain, and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in *Passive and Active Measurement Workshop (PAM2004)*, 2004.
7. M. Zukerman, T. D. Neame, and R. G. Addie, "Internet traffic modeling and future technology implications," in *IEEE INFOCOM*, 2003.
8. M. Li, M. Claypool, R. Kinicki, and J. Nichols, "Characteristics of streaming media stored on the internet," Tech. Rep. WPI-CS-TR-03-18, Computer Science Department, Worcester Polytechnic Institute, 100 Institute Road, Worcester, Massachusetts 01609-2280, May 2003.
9. J. Aracil, R. Edell, and P. Varaiya, "A phenomenological approach to internet traffic self-similarity," in *35th Annual Allerton Conference on Communication, Control and Computing*, 1997.
10. P. J. Brockwell and R. Davis, *Time Series: Theory and Methods.* New York: Springer, 1991.
11. R. Engle, "Autoregressive conditional heteroscedasticity and estimates of variance of UK inflation," *Econometrica*, no. 50, pp. 987–1008, 1982.
12. P. Embrechts, C. Kluppelberg, and T. Mikosch, *Modelling Extremal Events for Insurance and Finance.* Berlin Heidelberg: Springer, 1997.
13. A. Weiss, "Asymptotic theory for ARCH models: Estimation and testing," *Econometric Theory*, no. 2, p. 31, 1986.
14. J. A. Rice, *Mathematical Statistics and Data Analysis.* Belmont, California: Duxbury Press, 1995.
15. H. Hotelling, *Multivariate Quality Control. Techniques of Statistical Analysis.* McGraw-Hill, New York: Eisenhart C, Hastay MW, Wallis WA, 1947.

Reducing Large Internet Topologies for Faster Simulations

V. Krishnamurthy[1], M. Faloutsos[1], M. Chrobak[1], L. Lao[2],
J.-H. Cui[3], and A.G. Percus[4,⋆]

[1]U.C. Riverside
[2]UCLA
[3]U. Connecticut
[4]Los Alamos National Labs and UCLA IPAM

Abstract. In this paper, we develop methods to "sample" a small realistic graph from a large real network. Despite recent activity, the modeling and generation of realistic graphs is still not a resolved issue. All previous work has attempted to grow a graph from scratch. We address the complementary problem of shrinking a graph. In more detail, this work has three parts. First, we propose a number of reduction methods that can be categorized into three classes: (a) deletion methods, (b) contraction methods, and (c) exploration methods. We prove that some of them maintain key properties of the initial graph. We implement our methods and show that we can effectively reduce the nodes of a graph by as much as 70% while maintaining its important properties. In addition, we show that our reduced graphs compare favourably against construction-based generators. Apart from its use in simulations, the problem of graph sampling is of independent interest.

1 Introduction

Small graphs that resemble the Internet topology are needed for conducting simulations of various network protocols. Real graphs can have prohibitively large sizes, especially for highly detailed simulations such as packet level simulations. To produce high confidence results, one averages the experimental results over many graphs of a given size. Running the experiments over a range of sizes allows reseachers to interpolate the results to graph sizes outside the tested range. In particular, it shows whether the performance of the tested protocols scales well with increasing size, leading to accurate performance predictions for the Internet graphs of the future.

Currently, all known models for graph generation incrementally grow a graph with desired properties. Our work follows the opposite approach: we wish to reduce real large Internet instances to produce small realistic topologies. This task can be thought of as *graph sampling*, and it has attracted attention in other settings [18] [19].

Among the existing Internet topology generators, none has yet been widely accepted as sufficiently accurate. These generators produce arguably realistic graphs, but they do not necessarily match all the known topological properties of the Internet. Most graph generators attempt to grow a graph, an approach that we call **constructive**. This area has

[⋆] This work was supported by the NSF CAREER grant ANIR 9985195, and NSF grant IDM 0208950, and DARPA award FTN F30602-01-2-0535. NSF/NeTS 0435230. and NSF CCR 0208856 and funding from the LANL LDRD and UCDRD programs.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 328–341, 2005.

seen unprecedented activity since the discovery of skewed degree distributions in the Internet topology [11]. The generators either use "biased" or *preferential* growth policy [2] [3] [5] [10] [23] or force a power-law degree distribution [1] [16]. The weakness of these constructive methods lies in their dependence on the principles of construction, and the choice of parameter values. Furthermore, several of them focus on matching the degree distribution, while they often fail to match other topological properties, as multiply documented [5] [7] [15] [16] [29].

In this paper, we address the following problem: we want to "sample" a real topology[1] to produce a smaller graph. The overarching goal of our approach is very practical: we want the simulations on the sampled graph and the initial larger graph to lead to the same conclusions. This is a novel problem in the Internet modeling community, although some related work in other areas exists [18] [19]. We call our approach of generating a graph **reductive**. Intuitively, our approach has the easier task of not "destroying" the existing properties, in contrast to the task of the constructive approach, which has to reproduce all the right properties.

How do we evaluate the success of our approach? Establishing criteria for the realism of a generated graph is an open ended problem. In the case of graph sampling, the question is more involved: which Internet instance should the reduced graph try to match? One can distinguish two objectives: we can either try to match the properties of: (a) the real Internet instance of the same size (thus "reversing" the evolution of the Internet), or (b) the initial instance (thus producing its small imitation.) If the properties do not change with size, then both goals are equivalent. However, no obvious time independent topological metrics seem to exist [28]. The findings of [28] suggest that even though every Internet instance at the AS-level has power-law characteristics, there are variations in the value of the slope. Thus we chose the first method above, and we compare the reduced graph with an *equal size* real Internet instance.

The contribution of this paper is twofold: (i) we provide efficient graph sampling algorithms, and (ii) we compare our reduction methods against constructive methods. In addition, we compare our graphs using network protocol simulation but this study could not be included here due to space limitations. This paper significantly extends a preliminary version of this work [21], while a more detailed version appears as a technical report [20].

Graph Sampling Algorithms. As our main contribution, we develop and quantify the performance of a number of reductive methods. We group these methods into three main categories: (a) deletion methods, (b) contraction methods and (c) exploration methods. Our work yields the following results.

- Our best algorithms successfully reduce the graph size by up to 70%, in the number of nodes, while preserving the desired topological properties. Our methods are statistically robust to the initial topology and the randomization seed.
- We show analytically that some of our methods will maintain the power-law of the degree distribution, if such a distribution exists in the initial topology.

Comparison of Reductive and Constructive Methods. We compare our best reduction methods with commonly used constructive methods and find that our methods

[1] To make it more specific, the current Internet has more than 14,000 nodes. The smallest available Internet instance (from 1997) has about 3,000 nodes, but even this size is computationally expensive, if not prohibitive, for some types of simulations such as BGP simulations or flow level simulations [8] [27] [14].

match more closely the properties of the real instances. We find that the best constructive generator is Inet [16], which takes as input the available real instances. Inet currently does not generate graphs with less than 3000 nodes. Therefore, we can confidently say that for really small graphs ($n < 3,000$), the reductive methods are the best choice.

It is worth noting that the reductive approach has two additional attractive properties:

- A "statistically fair" reduction may preserve many graph properties, including some that we have not used for our metrics, or even some properties that we have not yet identified.
- The reductive method is likely to extend to different types of graphs, for example, the policy-based Internet topology, or the Web graph.

Graph sampling can be used as a tool to provide insight into the topological properties and structure of the graph. Finally, sampling can also complement a visualization effort, when the sizes are too large for a meaningful graphical representation.

2 Background and Metrics

In this section, we introduce the topology model and several topological properties of the Internet, which we use to evaluate the realism of our graphs.

2.1 Internet Instances

The Internet is divided into autonomously administered domains or Autonomous Systems (AS). In our study, we focus on the AS level topology, and we model the Internet as an undirected graph whose nodes are AS's and whose edges are inter-domain connections. This has been a standard approach in the past literature and there continues to be significant activity in measuring and modeling the AS level Internet as an undirected graph. It is worth noting, however, that, more recently, there have been efforts [13] [22] [4] to model the Internet as a directed graph by including business relationships.

Our real data come from the Oregon Routeviews project [12]. This is the frequently used archival data by researchers in this area and the only data archive that has instances spanning over 5 years. This data is specifically chosen for our study as we need a wide range in the size of the Internet topology. This was the reason why we could not use the [7] archive which spanned only over three months. Each instance in this paper is named using its collection date, in the format IYYMMDD. For example, the instance collected on May 07, 2001 is named I010507. We use real Internet instances [12] from November 1997 to March 2003 in our experiments.

2.2 Graph Properties

Several graph properties have been proposed to capture the characteristics of real Internet graphs [11] [15] [26], and we adopt most of these metrics in our study. Needless to say, preserving these properties is necessary for the generated graphs to resemble the Internet topologies, but it may not be sufficient, for these topologies may share other properties that have not yet been discovered.

Average and Standard Deviation of Degree. The average degree of a graph is equal to $2m/n$, where m is the number of links and n is the number of nodes. The average degree represents the density of a graph. The average degree of Internet topologies is known to increase over time, as the size of the graph also increases. Another measure we examine is the standard deviation of the degree distribution, which can be thought of as a measure of the "diversity" of the nodes in the network.

Degree Distribution. It has been shown that power laws approximate well[2] the skewed degree distribution [11]. Here, we focus on power law 1, the *degree rank exponent* and power law 2, the *degree exponent*. Degree rank exponent is defined as the slope of log-log plot of the nodes' degrees versus their rank, where the k-th ranked node is the one with the k-th highest degree. Degree exponent[3] is the slope of the log-log plot of the degree frequency versus degree. The two power laws can be shown to be, in fact, equivalent. In practive, however, the actual degree distributions follow these power laws only approximately, and studying both laws provides slightly different approximate views of the real degree distribution[4]. In this metric, we check the existence of power-laws and then compare the value of the exponent of the power laws [24]. Power laws are approximations whose accuracy is typically quantified by the correlation coefficient. In addition, power laws often target the tail of a distribution, which is the focus of our analytical work.

Spectral Analysis. Gkantsidis *et al.* [15] characterize the clustering and spatial properties of a topology using spectral analysis of the adjacency matrix of a graph. Spectral analysis captures significant information about the clustering properties of the topology in a unique way. It subsumes the clustering coefficient metric that was used before [5].

In more detail, spectral analysis examines the eigenvectors corresponding to the largest eigenvalues of the normalised transposed adjacency matrix of an entire topology. These vectors correspond loosely to the eigenvectors of the main clusters in the topology. The resulting plot depicts the 100 largest eigenvalues in order of magnitute, from largest to smallest. It is found that the clustering properties (the corresponding plot) have not changed significantly despite the Internet growth [15].

C. Graph Generators. Early graph generators failed to match the skewed degree distribution [6] [9] [31] [32]. Several recent generators build topologies with power-law degree distribution in mind [1] [3] [5] [16]. It is worth mentioning that the pioneering Barabasi-Albert model [3] generates a graph through preferential attachment: in attaching new nodes to existing ones, it favors high-degree nodes. Mitzenmacher provides an overview of methods to generate power law distributions [25].

To illustrate that the reduction methods generate graphs which resemble Internet topology better than the constructive methods, we compare the topology obtained by reducing the AS level Internet topology using our best reduction method with similar graphs generated by Inet [16], Waxman [30], Barabasi-Albert [3], and the modified GLP heuristic [5].

Finally, the problem of graph reduction and sampling appears in other disciplines, often with different goals. For example, graph sampling has been used in graph parti-

[2] Chen *et al.* [7] created a more complete Internet graph at the BGP level, but recent work by Siganos *et al.* [28] shows that the power laws hold with 99% correlation coefficient even in this new graph.

[3] We use the reverse cumulative distribution function (RCDF) of power law 2, which is more robust than the cumulative distribution function (CDF)[28].

[4] The correlation coefficient of the power law fit was verified by the authors of [7], who use more metrics to examine the goodness of the fit.

tioning in the context of distributed computing [18] [19], and randomized graph sampling has been used to solve different graph problems, such as min-cut approximation [17]. To our knowledge, however, prior to our work, no research has been reported on using graph sampling for generating realistic network topologies.

3 Graph Reduction Methods

This section presents our approach for reducing a real AS level Internet topology to a smaller realistic topology. Our methods fall into three categories: (a) *deletion methods*, that remove edges or nodes from the graph, one by one, until a desired size is reached, (b) *contraction methods*, that contract adjacent nodes, step by step, until a desired size is reached, (c) *exploration methods*, that traverse a desired number of nodes according to a given exploration policy, and retain the subgraph induced by those nodes. For consistency of notation, we abbreviate the methods starting with the letter that indicates the category they belong to: D for deletion, C for contraction, and E for exploration.

3.1 Deletion Methods

Our deletion methods are embedded in the following framework: The input consists of an initial graph G with n nodes and m edges, and the total percentage P of nodes to be deleted. The graph is reduced iteratively in stages, where at each stage a small percentage s of nodes is removed (where s is a parameter that can be set by the user.) A stage consists of several steps, in which we remove either one edge or one vertex selected according to the specific method. After each stage, connected components are found and the largest connected component is retained. The procedure stops when the reduced graph has approximately $n(1 - P/100)$ nodes. By reducing a small percentage s of the graph in each iteration, we are able to meet the target size more accurately. In practice, a reduction of 3% to 5% of the nodes at each stage was sufficient to achieve the desired reduction. (The partition into stages was introduced for efficiency, for maintaning connected components incrementally, under node or edge deletion operations, is either very slow or cumbersome to implement.) The deletion methods we study are:

Deletion of Random Vertex (DRV): Remove a random vertex, each with the same probability.

Deletion of Random Edge (DRE): Remove a random edge, each with the same probability.

Deletion of Random Vertex/Edge (DRVE): Select a vertex uniformly at random, and then delete an edge chosen uniformly at random from the edges incident on this vertex.

Hybrid of DRVE and DRE (DHYB-w): In this method, with probability w we execute DRVE and with probability $(1 - w)$ we execute DRE. In particular, DHYB-1 is DRVE, and DHYB-0 is DRE. (This method was motivated by our initial studies showing that DRVE and DRE had opposite performances with respect to different metrics, namely when one of them underestimated a metric's target value then the other overestimated it.) We consider nine values of w in our experiments, ranging from 0 to 1.0 in increments of 0.1. For clarity, we show only a subset of those here.

3.2 Contraction Methods

These methods proceed by contracting adjacent nodes. The two methods below differ in the manner their connecting edge is chosen.

Contraction of Random Edge (CRE): Pick a random edge, uniformly, and contract its endpoints. The neighbors of the merged nodes become neighbors of the new node. (This method bears some similarities to the random matching method [19] and the edge coarsening method [18]. We also considered a generalization of the CRE method, where more neighbors contract all at once, but the results were not satisfactory and are not shown here.)

Contraction of Random Vertex/Edge (CRVE): Pick a random vertex, uniformly, and contract it with a uniformly-chosen random neighbor.

3.3 Exploration Methods

Here, we pick an initial node randomly, traverse the graph according to a given exploration method, until a desired number of nodes is visited. We then retain the subgraph induced by these nodes: all nodes that have been visited and the edges between them are retained in the final graph. We study two ways to explore a graph:

Exploration by Breadth First Search (EBFS): Randomly select a start node, and then do breadth-first search starting from that node, until the desired number of nodes have been visited.

Exploration by Depth First Search (EDFS): Randomly select a start node, and then do a depth-first search starting from this node (following a random yet non-traversed edge at each forward step), until the desired number of nodes have been visited.

4 Analysis

In this section, we prove that two of our reduction methods, DRE and DRV, preserve the degree power-law. More specifically, we show that if an original graph satisfies the power law, then the reduced graph satisfies it too, with the same exponent, for large degrees.

Let G denote the original graph with n vertices and m edges. By n_d we denote the number of nodes of degree d, and by d_{ave} the average degree. These quantities are related to each other by $n = \sum_d n_d$, $m = \frac{1}{2}\sum_d dn_d$, and $d_{\text{ave}} = 2m/n$.

The symbols n', m', n'_d, d'_{ave} denote the corresponding values in the reduced graph G'. Since our reduction methods are probabilistic, these symbols actually represent expected values of the corresponding random variables.

We assume that the degree sequence of G satisfies the power law in the following form: $n_d = Cnd^{-\alpha}$, where $C = (\sum_{d=1}^{n} d^{-\alpha})^{-1}$ and α is the degree exponent. We wish to show that a similar property holds (approximately) in G'.

DRE and Power Law Preservation. The DRE method, as implemented in our experiments, removes edges at random, one at a time, and retains the largest connected component. This process, in its raw form, is not amenable to analytical studies, as the degree distribution of the eliminated nodes depends heavily on (unknown) topological properties of G. To facilitate the analysis, we will approximate DRE by another process that is easier to analyze. This approximation will proceed in several steps.

First, we ignore the fact that DRE removes the nodes outside the largest connected component, and study instead the degree distribution among *all* the vertices of G, after the edges are deleted. Thus, throughout this section, G' has the same vertex set as G and $n' = n$. This simplification is justified by experimental results showing that the nodes eliminated by DRE have very low degree, so this simplification should not affect the asymptotic behavior of the degree distribution.

Let $p = m'/m$. We can think of p as a probability of an edge being retained in the graph. Thus, in the second approximation, instead of removing edges one by one, we will flip a coin for each edge, independently, and remove each with probability $q = 1 - p$. Although this does not guarantee that the resulting graph will have exactly m' edges, its expected number of edges is very heavily concentrated around m', and the two processes have asymptotically the same behaviors.

Informal argument. Our goal is to show that in G' the degrees satisfy $n'_d = C'n'd^{-\alpha}$, for some constant C'. The general idea of our proof is summarized as follows. Roughly speaking, nodes in G with degree between $(d - \frac{1}{2})/p$ and $(d + \frac{1}{2})/p$ end up in G' with an expected degree of d. Since this range covers $1/p$ different degrees, and for degrees c close to d the values n_c are close to n_d, we might anticipate that for d not too small, we should get $n'_d \sim \frac{1}{p} n_{d/p} = Cp^{\alpha-1}n'd^{-\alpha}$, preserving the power law with the same exponent.

Theorem 1. *For any fixed exponent $\alpha > 1$ and probability $p \in (0, 1)$, given a graph with degree distribution given by $n_d = Cnd^{-\alpha}$, the degree distribution of the graph reduced through the process above will approximately follow a power-law: $n'_d \approx Cp^{\alpha-1}$ $d^{-\alpha}$.*

The formal statement of the above theorem and its proof are omitted due to space limitations.

DRV and Power Law Preservation. An analogous argument as for DRE can also be applied to DRV. We only outline an informal explanation here: Let $n' = pn$. We can think about DRV as removing each vertex in G, independently, with probability $q = 1 - p$. Then, roughly speaking, a fraction p of the nodes with degree between $(d - \frac{1}{2})/p$ and $(d + \frac{1}{2})/p$ end up in G' with an expected degree of d. Other nodes are either deleted or their new degrees are not d. Since this range covers $1/p$ different degrees, and for degrees c close to d, the values n_c are close to n_d, as long as d is large enough, we should get $n'_d \sim n_{d/p} = Cp^{\alpha-1}n'd^{-\alpha}$, preserving the power law with the same exponent.

5 Graph Reduction Evaluation

We examine the performance of our sampling methods in practice. The starting point of the reduction in most of the experiments in this paper is the AS level Internet topology I010507 collected on 07/05/2001. (However, we have experimented with other topologies with similar results.) The I010507 graph has 10,966 nodes and 22,536 edges, thus an average degree of 4.11. The "Internet curve" shown in all the graphs represents the value of actual Internet instances of size corresponding to the value on the x-axis. If we reduce the I010507 graph by 70% we end up with about 3,300 nodes which is roughly the same size as the I980124 graph. Each data point in our plots represents the *average of 50 runs* with different random seed.

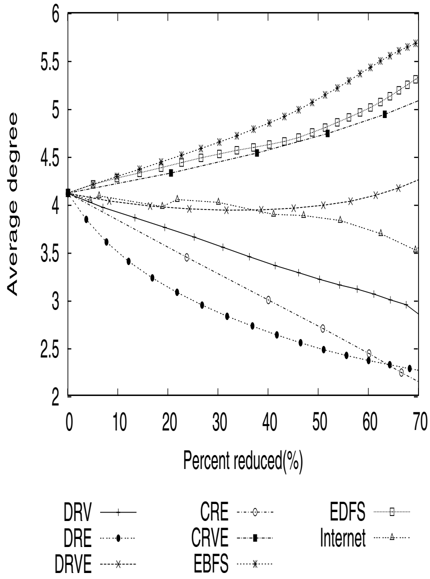


Fig. 1. Average degree comparison of Deletion, Contraction and Exploration Methods

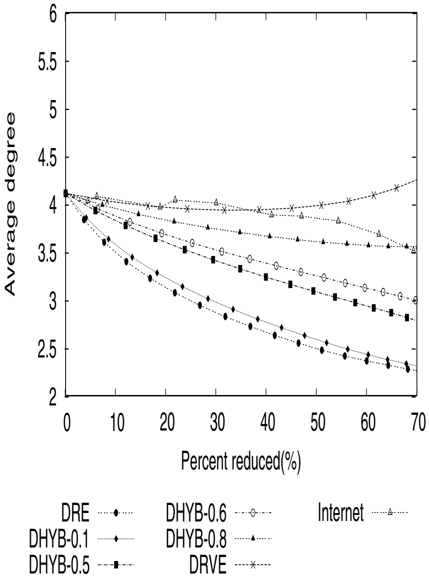


Fig. 2. Average degree comparison of Hybrid Methods

Our experimental results show that among all the methods DHYB-0.8 seems to have the least deviation from the Internet's topological properties. Thus it is the best method with respect to topological metrics described in Section II. Among the non-hybrid methods, DRV performs well. Recall that the DHYB method combines edge deletions DRE and DRVE. It is interesting to see how well the random node removal DRV worked in practice. In Table 1 we present the top performing methods according to some metrics. Variations of DHYB and DRV are consistently present in every column. We have not shown the performance of all the DHYB methods here as the graph becomes very congested. As the value of w in DHYB is increased, the metric values increased between DRE and DRVE.

Test 1: Average Degree and its Deviation. *DHYB-0.8 has the best performance.* Figure 1 shows how the average degree varies for the deletion, contraction and exploration methods. Figure 2 shows the average degree of the hybrid methods with $w = 0.1, 0.5, 0.6, 0.8$. DRVE follows the evolution of the average degree fairly closely up to 50% reductions, but then it diverges quickly. DHYB-0.8 stays close to the Internet curve in the whole range, and it has nearly the same value of average degree at the 70% reduction point.

We also found that DHYB-0.8 is better in terms of the average deviation, with an average percentage deviation of 4.2%, followed by DRVE with 5%. These methods are followed by DHYB-0.6 and DRV with average percent deviations of 11% and 12.2% respectively. We have selected only methods whose average degree decreased under graph reduction, mirroring the trend in the real Internet data observed in Figures 1 and 2. With the exception of one data point, the Internet's average degree constantly decreased

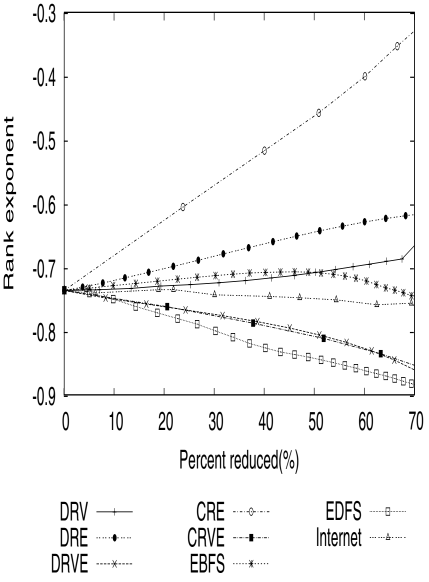


Fig. 3. Rank exponent comparison of Deletion, Contraction and Exploration Methods

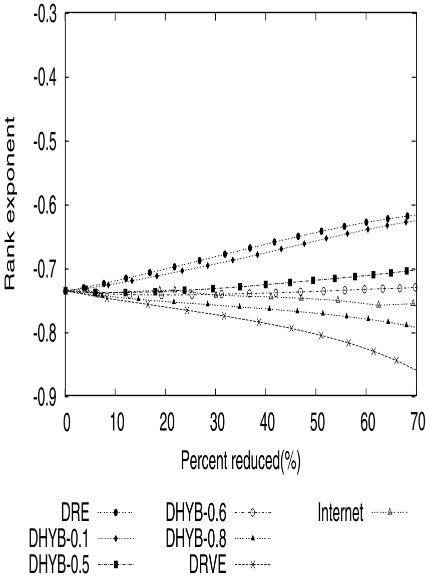


Fig. 4. Rank exponent comparison of Hybrid Methods

with decreasing size. All the other methods are farther away from the Internet, so we conclude that they do not fare well in this metric comparison.

Test 2: Exponent of Rank Power Law. *DHYB-0.6 is the best method.* The hybrid methods follow the variation of the Internet rank exponent more closely than the other methods, as is evident in Figures 3 and 4. In fact, the average percent deviations was within 3% for DHYB-0.6, DHYB-0.5 and DHYB-0.8. DHYB-0.8 and DHYB-0.5 are the second best methods after DHYB-0.6 which lie above and below the Internet line. DRV was quite close, with an average percentage deviation of 5.2%.

Test 3: Correlation Coefficient of Rank Power Law. *DHYB-0.5, 0.6, 0.8 and DRV consistently maintained a high correlation coefficient.* In addition to having an exponent value closer to that of the Internet, the methods should also have a high correlation coefficient, preferably above 97%. Even though it looks like EBFS performs equally well as DHYB-0.6, it has a smaller correlation coefficient (below 96%). A similar trend is seen in CRVE which follows DRVE very closely. The other methods have correlation coefficient above 96% except CRE whose correlation coefficient drops steadily from 90% (at 25% reduction) to 61% (at 70% reduction). Even though we include EBFS, CRVE and CRE in Figure 3 for degree exponent comparison, we exclude them from being viable solutions at this point.

Test 4: Exponent of Degree Power Law. *DHYB, DRV, and DRE are successful in this test:* their degree exponent is within 5.5% from the the exponent of the Internet instance. Among the hybrid methods, DHYB-0.5, 0.6 and 0.8 perform well, having a value within

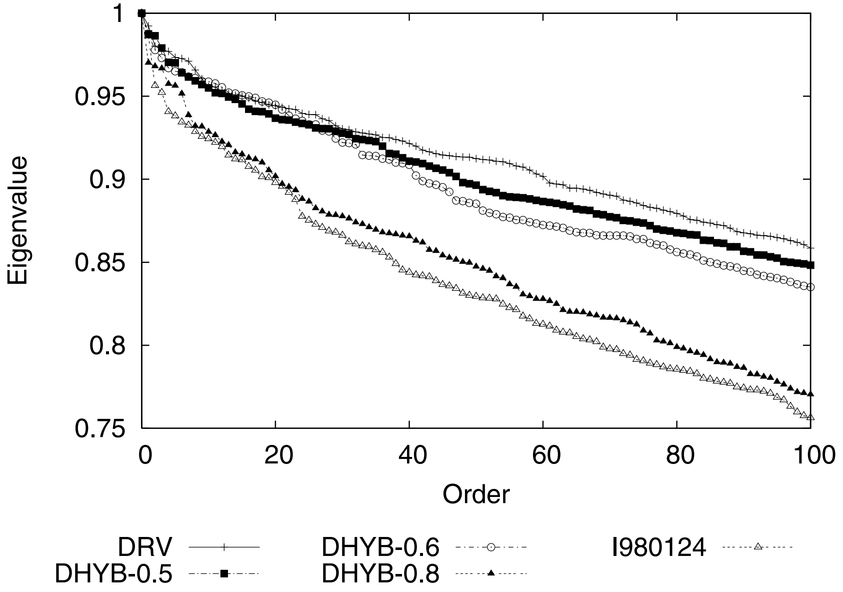


Fig. 5. Spectral Analysis of 70% reduced DHYB-0.5, 0.6, 0.8, DRV, and Internet instance I980124

Table 1. The best methods, based on average percent deviation from the target value. For average degree, "best" is within 5% and "second best" within 5-15%. For the other two metrics, "best" is within 3%, and "second best" within 3-5.5%

Range	**Average Degree**	**Rank Exponent**	**Degree Exponent**
Best group	DRVE,DHYB-0.8	DHYB-0.1,0.6,0.5,0.8	DHYB-0.1,DRE, DHYB-0.5
Second group	DRV,DHYB-0.6,0.5	DRV	DHYB-0.6,0.8,DRV

or close to 5% of the exponent of the Internet topologies. (Figures not included due to space limitations.)

Test 5: Correlation Coefficient of Degree Power Law. *The correlation coefficient of the best methods namely DHYB-0.1,0.5,0.6,0.8, DRE and DRV are above 97% in all the cases.*

We evaluate the methods based on their average percent deviations from the Internet with respect to the four metrics we examined so far. From Table 1, we conclude that DHYB-0.8, 0.6, 0.5 and DRV are the leading methods, and from now on we will use only those four methods in the remaining experiments.

In the following two tests, we need to generate a plot for every topology, unlike the previous tests where we had a single value corresponding to a topology. Thus we chose to show only the 70% reduction point; we had similar results for the other reduction points also. We could maintain successfully the above mentioned topological properties up to the 70% reduction point using our methods: DHYB-0.5, 0.6, 0.8 and DRV. For the hop-plot and spectral analysis test, we reduce the I010507 topology by 70% using DHYB-0.5, 0.6, 0.8 and DRV. The reduced graph now has about 3290 nodes and its performance is compared with the I980124 Internet topology having 3291 nodes.

Test 6: Spectral Analysis. *DHYB-0.8 gives best results.* The spectral analysis results of DHYB-0.8, 0.6,0.5 and DRV (the selected best methods) are shown in Figure 5. Recall that the spectral behavior of the Internet topology is consistent over time [15]. So we have reason to believe that the reduction method whose spectral behavior matches I980124 is the best method. As we can see DHYB-0.8 follows the I980124 topology closely, outperforming the other methods.

Considering the results of all tests above, the best of the methods we tested is DHYB-0.8, as it maintains an average percent deviation from the target values close to or below 6% with respect to the four topological metrics, and in several tests it outperforms all other methods. Among the non-hybrid methods, DRV seems to the best method maintaining an average percent deviation close to or below 5% for the power-law metrics and within 15% for average degree.

Test 7: Robustness to Input Internet Instance. *All the methods are insensitive to the initial instance.* We further investigated the stability of each method with respect to the input Internet instance. We tested our seven reduction methods on the most recent AS level Internet topology I030313 with 15,026 nodes and 31,200 edges. The results were very similar to those reported for the instance I010507. Similar results were also obtained for other Internet instances.

6 Reductive Methods versus Constructive Methods

We compare our reduction methods with existing well-known constructive generator: Inet [16], Waxman [30], BA (Barabasi-Albert) [3], and GLP (Generalized Linear Preference) [5]. Using the same metrics as in Section 5, we compare the topologies reduced by our best reduction methods, namely DHYB-0.8 and DRV (starting from instance I010507) with topologies from these other topology generators. For brevity, we show results only for two selected metrics.

Test 8: Average Degree. *Inet follows closely the variations in the Internet's average degree.* The behavior of Inet is not surprising as this generator predicts the average degree using real Internet instances from the same data archive [12] that we use, and forces this degree distribution. DHYB-0.8 is the next best method. DRV doesn't follow the variations in the Internet but decreases in value linearly, unlike GLP which varies haphazardly with no specific pattern. The BA and Waxman generators produce topologies with an average degree of 4 independent of the size of the graph.

Test 9: Exponent of Rank Power Law. *DHYB-0.8 is the best method.* The hybrid method follows the variation of the Internet rank exponent very closely as is evident in Figure 6. We recall from the previous section that the average percent deviation of DHYB-0.8 with respect to this metric was within 3%. Inet maintains a constant value for the exponent irrespective of the size of the graph. DRV has values higher than the Internet and is the next best method. In the BA generator, both the node placement options (random and heavily tailed) generate topologies with similar values. Similar to the previous test, the exponent value is independent of the size for both Waxman and BA topologies.

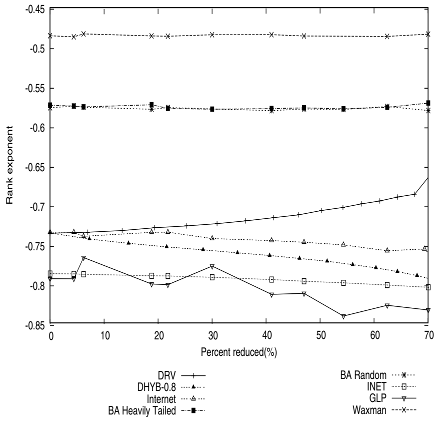


Fig. 6. Rank exponent comparison of Reductive and Constructive Methods.

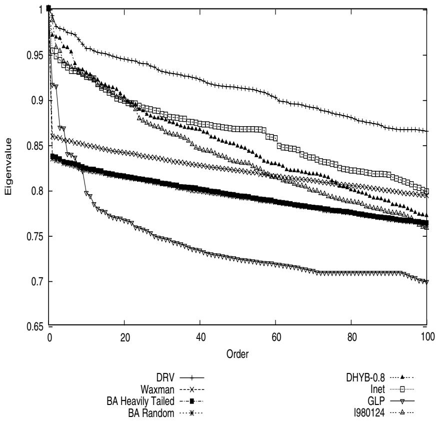


Fig. 7. Spectral Analysis of Reductive and Constructive Methods

Test 10: Exponent of Degree Power Law. *DYB-0.8 and DRV seem to be best.* Most of the methods perform well, except for those by Waxman and BA. In particular, DHYB-0.8 and DRV have values close to the Internet. They are followed by GLP and Inet, which fall above and below the Internet respectively.

Test 11: Spectral Analysis. *DHYB-0.8 seems to be the best.* Synthetic generators like BA and GLP have not only smaller eigenvalues compared to the Internet but also a slope value that is very different from the AS level Internet topology [15]. Gkantsidis *et al.* [15] claim that these generators fail to reproduce the strong clusters that are present in the Internet. On the other hand, our methods DHYB-0.8 and DRV have a higher eigenvalue and a distribution very similar to the Internet (Figure 7). The eigenvalues of Inet doesn't decrease gradually unlike the Internet but instead exhibits sharper trends.

Summary. We find that DHYB-0.8 is the best method followed by Inet and DRV. Inet, however, does not generate graphs below 3000 nodes. (This could be related to the fact that Inet uses the available instances from the RouteViews archive [12] in order to calibrate its intended graph metrics, and the smallest instance (collected on 15th Nov 1997) in the archive has 3,037 nodes.) Given this restriction, we believe that DHYB-0.8 is the best choice for small graphs ($nodes < 3,000$). We used such small topologies, including the I980124 graph reduced to 1500 nodes, to evaluate our generation techniques in multicast simulations. We show that even using such small graphs, we can obtain realistic simulation conclusions. (Results not included due to space limitations.)

7 Conclusion

The goal of this paper has been to propose and study methods for sampling Internet-like graphs. We propose and evaluate the performance of three types of reduction methods with multiple methods of each type. Our work leads to the following conclusions.

How can I sample a real network? We conclude from our experiments that DHYB-0.8 is the best among our methods for the Internet sampling, and that it also compares favorably to graph generation methods proposed previously in the literature. DRV is nearly as good, which, given DRV's simplicty, is an interesting result in its own right.

How much can I reduce a real network? We are able to reduce a graph successfully by approximately 70% in terms of the number of nodes. Beyond 70% we often find that the statistical confidence coefficient is low.

Provable reduction performance. We show analytically that DRV and DRE respect an initial power-law degree distribution.

Simulation speedup. The speedup depends on the complexity of simulations. Given a 70% reduction in size, an $O(n^2)$ or $O(n^3)$ simulation will decrease by a factor of about 11 or 37, respectively. Furthermore, smaller graphs will require less memory which can decrease the simulation time further.

References

1. W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. *STOC*, 2000.
2. R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Review of Modern Physics*, 2002.
3. A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 8, October 1999.
4. G. Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. *IEEE INFOCOM*, 2003.
5. T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. *Infocom*, 2002.
6. K. Calvert, E. Zegura, and M. Doar. Modeling Internet topology. *IEEE Trans on Communication*, pages 160–163, December 1997.
7. Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. The origin of power laws in Internet topologies revisited. *INFOCOM*, 2002.
8. X. A. Dimitropoulos and G. F. Riley. Creating realistic BGP models. *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2003.
9. M. Doar. A better model for generating test networks. *Proc. Global Internet, IEEE*, Nov. 1996.
10. A. Fabrikant, E. Koutsoupias, and C.H.Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the internet (extended abstract). *STOC*, 2002.
11. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM*, pages 251–262, Sep 1-3, Cambridge MA, 1999.
12. National Laboratory for Applied Network Research. Online data and reports. Supported by NSF, http://www.nlanr.net, 1998.
13. Lixin Gao. On inferring automonous system relationships in the Internet. *In Proc. Global Internet*, November 2000.
14. G.F.Riley. On standardized network topologies for network research. *Simulation Conference, 2002. Proceedings of the Winter*, 1:664 –670, 2002.
15. C. Gkantsidis, M. Mihail, and E. Zegura. Spectral analysis of Internet topologies. *IEEE INFOCOM*, 2003.
16. C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. *Techical Report UM CSE-TR-433-00*, 2000.

17. D. Karger. Randomization in graph optimization problems: A survey. *Optima*, 58:1–11, 1998.
18. G. Karypis. Multilevel hypergraph partitioning. *Technical Report, Department of Computer Science, University of Minnesota: 02-025*, 2002.
19. G. Karypis and V. Kumar. A fast and high quality scheme for partitioning irregular graphs. *Technical Report, Department of Computer Science, University of Minnesota: 95-035*, 1995.
20. V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J.H. Cui, and A. G. Percus. Reducing large internet topologies for faster simulations, 2005. UC Riverside, Technical Report.
21. Vaishnavi Krishnamurthy, Junhong Sun, Michalis Faloutsos, and Sudhir Tauro. Sampling internet topologies: How small can we go? In *International Conference on Internet Computing, Las Vegas*, 2005.
22. L.Subramanian, S.Agarwal, J.Rexford, and R.Katz. Characterizing the Internet hierarchy from multiple vantage points. *Proc. IEEE INFOCOM*, 2002.
23. A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite:an approach to universal topology generation. *MASCOTS*, 2001.
24. A. Medina, I. Matta, and J. Byers. On the origin of powerlaws in Internet topologies. *ACM SIGCOMM Computer Communication Review*, 30(2):18–34, April 2000.
25. M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2), 2004.
26. C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. *SIGKDD*, 2002.
27. G. F. Riley, M. H. Ammar, R. M. Fujimoto, K. Perumalla, and D. Xu. Distributed network simulations using the dynamic simulation backplan. *International Conference on Distributed Computing Systems 2001 (ICDCS'01)*, 2001.
28. G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power-laws of the Internet topology. *IEEE/ACM Trans. on Networking*, August 2003.
29. H. Tangmurankit, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. Network topology generators: Degree-based vs structural. *SIGCOMM*, 2002.
30. B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.
31. E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an Internetwork. *IEEE INFOCOM*, 1996.
32. E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for Internetworks. *TON*, 5(6), December 1997.

Dimensioning the Contention Channel of DOCSIS Cable Modem Networks

J. Lambert, B. Van Houdt, and C. Blondia

University of Antwerp, Department of Mathematics and Computer Science,
Performance Analysis of Telecommunication Systems Research Group,
Middelheimlaan, 1, B-2020 Antwerp – Belgium
{joke.lambert, benny.vanhoudt, chris.blondia}@ua.ac.be

Abstract. This paper presents an open queueing model with blocking that enables us to determine the optimal fraction c^* of the uplink channel capacity that should be dedicated to the contention channel in order to minimize the mean response time in a DOCSIS cable network. To assess the performance of this queueing network we make use of a decomposition technique. One of the key components of the model exists in capturing the behavior of the conflict resolution algorithm by means of a single processor sharing queue. The model is validated in three successive steps by means of several simulation programs. We also explore the impact of a variety of systems parameters, e.g., the number of cable modems, the initial backoff window size, etc. , on the optimum c^*.

Keywords: DOCSIS Cable Networks, contention channel, performance evaluation, queueing networks, decomposition algorithm.

1 Introduction

Recently, the rapid growth of the number of residential Internet users and the increased bandwidth requirements of multimedia applications have necessitated the introduction of an access network that can support the demand of such services. The Data Over Cable Service Interface Specifications (DOCSIS) [3] are the dominant specifications for carrying data over Cable TV Distribution (CATV) networks and have been developed by CableLabs and MCNS (Multimedia Cable Networks Systems), which is a group of major cable companies, to support IP flows over Hybrid Fiber Coaxial (HFC) networks. DOCSIS defines modulation and protocols for high speed bi-directional data transmissions over cable systems. It has been accepted by most major vendors and is now a widely used specification to provide high-speed residential access. DOCSIS specifies a set of interface protocols between the cable modem customer side and the termination network side.

The Media Access Control (MAC) protocol defined in the DOCSIS RFIv1.1 is based on Time Division Multiple Access (TDMA). It uses MAC management messages, referred to as MAP messages, to describe the usage of the uplink channel. A given MAP message indicates the upstream bandwidth allocation

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 342–357, 2005.

over the next MAP time, termed the *MAP length*. The MAP assigns some uplink minislots to particular cable modems (CMs) to transmit data, other slots are available as contention slots to request bandwidth. This is one of the critical components of the DOCSIS MAC layer and the DOCSIS specification purposely does not specify these bandwidth allocation algorithms so that vendors are able to develop their own solutions. In this paper, we develop an open queueing network with blocking, whose performance determines the optimal ratio between contention slots and reservation slots in a single MAP length.

The paper is structured as follows. Section 2 presents a general description of the cable network considered. The contention resolution algorithm specified by the DOCSIS standard is discussed in Section 3. The queueing network model together with the decomposition techniques used to assess its performance is introduced in Section 4, whereas Section 5 validates this model in three successive steps. Finally, we numerically explore the influence of several system parameters in Section 6.

2 The DOCSIS Cable Network

The DOCSIS cable modem network considered in this paper is shown in Figure 1 and consists of a single Cable Modem Termination System (CMTS), located in the head-end of a cable operator or service provider and a number of Cable Modems (CMs) that are installed at the end-users. At initialization time each CM registers itself with the CMTS and at least 2 service flows are created for each CM: one in the downstream and one in the upstream direction. Since the head-end is the only transmitter in the downstream channel, no downstream media access control (MAC) mechanism is needed. The upstream channel, on the other hand, is shared by a number of CMs and transports signals from the CMs to the head-end. The available bandwidth is divided into fixed length allocation units, called minislots, and the Data Over Cable Service Interface Specifications (DOCSIS) specify a reservation-based, centralized approach for distributing these minislots among the CMs.

Periodically, the CMTS sends a bandwidth allocation map (MAP) message over the downstream channel. A MAP message contains a number of data grants, such a grant indicates when a particular CM is allowed to transmit data on the uplink channel. Moreover, the MAP also identifies the minislots part of the contention channel. The total time scheduled in a single MAP message is referred

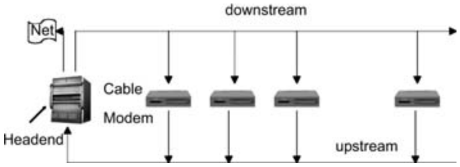


Fig. 1. Logical topology of a cable modem network

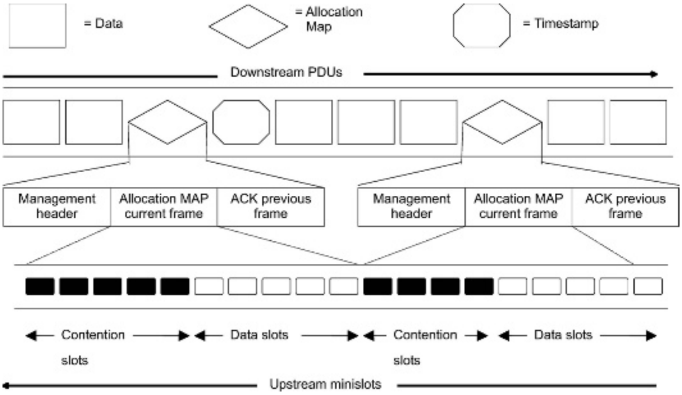


Fig. 2. Upstream bandwidth allocation

to as the MAP length and the time interval assigned to the contention channel is called the contention region, whereas the remaining part of the MAP length is termed the reservation region. Figure 2 illustrates the upstream mapping.

Any CM that wants to transmit data must request bandwidth to the CMTS by means of a request message that contains a count of the number of minislots needed. The request can be transmitted in 2 ways: (1) On the one hand the CM can send the request in the contention region. In this interval collisions may occur, meaning the request might get lost. The contention resolution algorithm (CRA) specified by the DOCSIS standard and operating on the contention channel, will be discussed in Section 3. (2) On the other hand, once a CM has received a grant, it has the opportunity to piggyback the new request in its reserved upstream allocation window, avoiding any collisions. Thus, piggybacking allows stations to request new bandwidth when transmitting data packets, without reentering the contention based request process. After processing a request, the CMTS generates a data grant and transmits it in the next MAP if capacity is available.

It is the objective of this study to determine how to distribute the minislots part of the MAP length between the contention and the reservation region in an optimal way, and to examine the influence of various system parameters, e.g., the number of CMs, on the location of this optimum. The obtained results apply to any DOCSIS cable network whose bandwidth allocation algorithm fairly distributes the upstream bandwidth between the requesting CMs, and given the mean packet length, are also independent of the data packet length distribution at hand.

3 The Contention Resolution Algorithm

This section discusses the collision resolution algorithm, being a truncated binary exponential backoff algorithm (BEB), specified by the DOCSIS standard. The aim of the BEB is to minimize the collision probability between the request packets transmitted in the contention region. Therefore, each CM has to postpone every transmission attempt by a random time interval. The length of this time interval, called the backoff interval, indicates the number of contention slots[1] that the CM must let pass before it transmits its own request and grows as the number of consecutive unsuccessful transmissions increases. That is, at each attempt to transmit a request packet, the length of the backoff interval is uniformly chosen in the range $[0, w-1]$, where w is called the current contention window (CW). The value of the parameter w depends on the number of transmission failures that occurred so far for the particular request packet. At the first transmission attempt, the CM initializes its backoff counter to 0 and w is set equal to $CW_{\min}$, called the minimal CW. After transmitting the request packet with a backoff counter equal to $i-1$, for $i > 0$, the CM waits for an ACK. The contention resolution phase ends if the request is successfully received by the CMTS, otherwise the CM increases his backoff counter (to i) and the contention window is set equal to $W_i = 2^{\min(i,m')} CW_{\min}$, where $CW_{\max} = 2^{m'} CW_{\min}$ is the maximum length of the CW. Thus, as soon as the maximum value $CW_{\max}$ is attained, the CW remains equal to its maximum value until there is either a successful transmission or until the maximum backoff counter value $m+1$ is reached. When the backoff counter is increased to $m+1$, the CM drops the request packet. The DOCSIS standard specifies that the maximum number of retries m should be set to 15. The values of $CW_{\min}$ and $CW_{\max}$ are not specified by the standard and are controlled by the CMTS.

In order to construct a queueing model to dimension the contention channel in a DOCSIS cable network, it is vital to capture the behavior of the BEB algorithm accurately. Although the BEB is considered as hard to model mathematically, Bianchi [2] managed to develop a fairly simple mathematical framework that provides accurate results for the BEB in an 802.11 setting. This model was further extended by a number of authors, including [8] and [6]. Although we are not considering the BEB in a 802.11 setting, we will demonstrate that nearly the same approach can be taken to accurately model the BEB in a DOCSIS world. A fundamental role in developing a model for the BEB is played by the saturation throughput $S(n)$. $S(n)$ is defined as the limit reached by the throughput of a system with n stations as the offered load increases and represents the maximum load that a system with n stations can carry under stable conditions. The saturation throughput values $S(n)$ are used in Section 4.1 to develop a processor sharing (PS) type of queue to model the contention channel. The computation of $S(n)$ is analogue to the approach taken by Bianchi [2] and Wu [8] and is therefore presented in Appendix A.

4 A Queueing Model for a DOCSIS Cable Network

To enable a mathematical analysis of the performance of DOCSIS cable networks, we will develop an open queueing network with blocking composed of

[1] The size of a contention slot, expressed in minislots, depends on the modulation scheme and equals the time needed to transmit a single request packet.

$N + 2$ queues, where N represents the number of CMs connected to the access network. The transmission buffer of each CM is represented by a single queue, whereas the remaining two queues represent the contention and the reservation channel, respectively. Details on the service disciplines of the contention and reservation queues are given after the general model description.

A data packet that needs to be transmitted from CM i toward the network is placed in the transmission buffer of CM i. If the packet finds the buffer empty upon arrival, CM i needs to generate a request for this data packet and has to send it to the CMTS via the contention channel, before the actual data transmission can take place. On the other hand, if some of the earlier data is still stored within the transmission buffer, we can use a piggybacking strategy to transmit the request collision free. In our queueing model, we represent each data packet by a customer. Depending on the progress of the packet transmission, its corresponding customer C is in one of the following 3 queues:

1. Customer C is waiting in queue CM i, whenever there are still other (older) packets that need to be (partially) transmitted first by CM i.
2. Customer C is part of the contention queue, if the data packet found the transmission buffer empty upon arrival and the CM is trying to transmit a request via the contention channel.
3. Customer C is part of the reservation queue, if all other (older) packets have been transmitted and a request for this packet was either piggybacked or successfully transmitted using the contention channel.

A new packet arrival at CM i corresponds to a new customer arrival in queue CM i, whereas customers who complete service in the reservation queue will leave the queueing network. Finally, a customer leaving the contention queue, will enter the reservation queue. Notice, the contention and reservation queues hold at most one customer per CM at a time. Thus, a customer at the head of line in a CM queue is blocked until the previous customer generated by the same CM has left the queueing system.

The reservation channel is modeled as a processor sharing (PS) queue, reflecting the DOCSIS MAC design principle of distributing the transmission capacity fairly among the active stations. The rate ν of the server is determined by (i) the rate of the uplink channel r (expressed in bits/msec), (ii) the mean length of a data packet L (including overhead) and (iii) the fraction $1 - c$ of the MAP length dedicated to the reservation region. That is, $\nu = (1 - c)r/L$. The contention channel is also modeled as a PS queue, but the service rate $\mu(n)$ depends upon the number of customers n present in the contention queue. The rate $\mu(n)$ is chosen as $cS(n)r/Q$, where c denotes the fraction of the MAP length associated with the contention region, Q the request size expressed in bits and $S(n)$ the saturation throughput. This is a fairly logical choice as $1/S(n)$ is also the mean time between two successful transmissions in a saturated system with n stations. The accuracy of this model for the contention channel is validated in Section 5.2. Using the saturation throughput as the service rate of a PS queue when modeling a contention channel is a proven technique (within the 802.11

setting), e.g., [4,6]. Finally, we assume that the packets arrive at the CMs according to a Poisson process with rate λ (expressed in packets/msec). In practice, the traffic at the CMs is likely to be more bursty and correlated in comparison with the Poisson arrivals, however, it should be noted that when dimensioning the contention channel, assuming Poisson arrivals will result in a pessimistic prediction for the amount of contention channel needed, because the piggybacking mechanism is more effective in dealing with bursty arrivals. Hence, we can use the optimal fraction c^* obtained for the Poisson arrivals as an upperbound for real world traffic scenarios. The current model can, without too much effort, also be extended to more bursty arrival processes like the batch Poisson process or a Markovian arrival process (MAP).

In general, queueing networks with blocking typically do not have a closed-form solution and therefore it is very difficult to analyze the system as a whole. In order to solve this queueing model, we make use of a decomposition scheme similar to [7]. The general approach in decomposition is to decompose the entire system into smaller subsystems, analyze these subsystems individually and then take into account the interactions between the various subsystems in putting them back together. We will make use of two subsystems, described in the next subsections, where the analysis of each subsystem requires information that can be obtained by solving the other subsystem.

4.1 The First Subsystem: A Closed Queueing Network

This subsystem is obtained by removing the waiting rooms in front of each of the N CM queues and by closing the network (see Figure 3). The resulting network is of the BCMP type (see [1]) with 3 service centers and N identical customers. Service center 2 (top right) represents the contention channel, whereas service center 3 (bottom right) represents the reservation channel. Service center 1 (left) has an infinite server (IS) queueing discipline, i.e., the number of servers in the service center is greater than or equal to the number of customers N in the network. A customer arriving at service center 1 always finds a free server and never queues for service. The service rate of service center 1 is equal to the arrival rate λ of packets at each CM, meaning a customer will on average stay

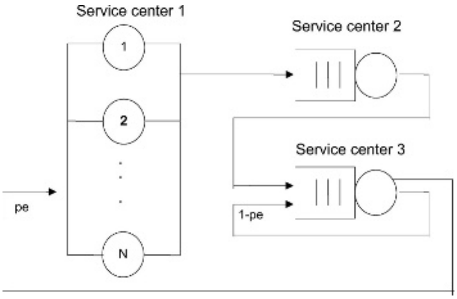


Fig. 3. The first subsystem

within service center 1 for $1/\lambda$ msec (this corresponds to the mean time needed for a new packet to enter an empty transmission buffer). Service centers 2 and 3 are identical to the Processor Sharing (PS) disciplines of the original queueing network. Notice, the packet length distribution is irrelevant (apart from its mean) as the performance measures of such a BCMP network are insensitive to the service time distribution of its PS service centers.

The routing matrix of this closed queueing network is equal to

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ p_e & 0 & 1-p_e \end{bmatrix}, \tag{1}$$

where p_e represents the probability that a packet finds the transmission buffer empty upon arrival. The value of p_e is determined by solving the second subsystem. Notice, p_e can be regarded as the probability that an arbitrary data packet leaves an empty transmission buffer behind, therefore, the CM buffer becomes empty for some time (with mean $1/\lambda$). To calculate the mean response times $E[R_{\text{cont}}]$ and $E[R_{\text{res}}]$ of service center 2 and 3 of this closed queueing network, one may use either one of the following algorithms [5]: the Convolution, the Mean Value Analysis (MVA) or the Local Balance algorithm. These mean response times are used as input values in the second subsystem.

4.2 The Second Subsystem

The second subsystem consists of N independent finite single server queues, one for each CM. The packets arrive at server i according to a Poisson process with rate λ, expressed in packets/msec, and the service discipline is FCFS. The service time at each server accounts for the time spent on the reservation channel (to transmit the data packet) and the possible time needed to visit the contention queue (to send the request), therefore its rate μ_{CM} is chosen as $\mu_{CM} = \frac{1}{E[R_{\text{cont}}]p_e + E[R_{\text{res}}]}$, where the mean response time of the contention channel, respectively the reservation channel, was denoted by $E[R_{\text{cont}}]$, respectively $E[R_{\text{res}}]$, and these values are obtained by solving the first subsystem.

The queue is finite and has a capacity of B customers, i.e., $B-1$ in the waiting room and one in the server. New arrivals are lost whenever the queue contains B customers. Hence, queue i can be solved as an $M/M/1/B$ queue. For this subsystem we can easily determine the steady state probabilities, from which we compute the probability p_e that the server is found empty by a new arrival. This probability is used as the new input value of p_e in the first subsystem.

The decomposition algorithm executes successive iterations during which both subsystems are solved one after the other, until the probability p_e has converged. That is, as soon as the absolute difference of p_e between two successive iterations is less than $\epsilon = 10^{-14}$. After the algorithm has converged, the probability p_e is used to calculate the total mean response time, which equals $E[R_{\text{cont}}]p_e + E[R_{\text{res}}]$, and the loss probability p_{loss}.

5 Model Validation

Our model is validated in three consecutive steps. First, we investigate the accuracy of the saturation throughputs $S(n)$ in a DOCSIS setting. This is done by comparing the analytical results obtained in Appendix A with a simulation of the BEB under saturated conditions. Second, we compare the performance of the contention queue with the performance of the BEB under Poisson arrivals and a finite population of size N. Finally, we simulate the original open queueing network with blocking to demonstrate the precision of the decomposition method.

5.1 The Saturation Throughputs $S(n)$

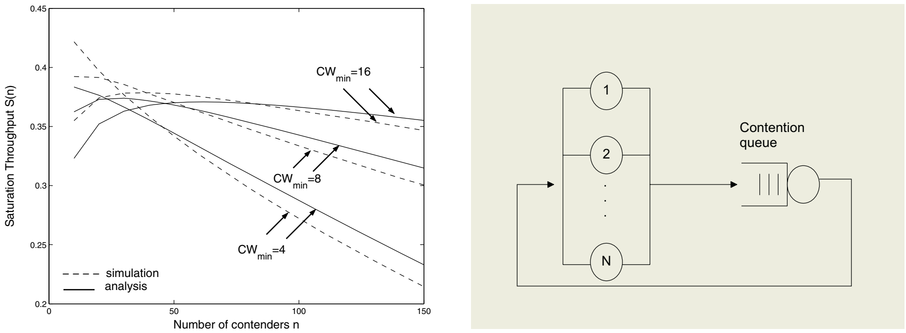
In Appendix A we presented an analytic evaluation of the saturation throughput. Although, [2] and [8] have validated the analytical saturation throughputs for the basic access and RTS/CTS access mechanisms in an 802.11 environment, we validate this model for a DOCSIS setting by comparing its results with those obtained by a simulator. We feel that this is necessary as the packet lengths, empty slot size and the collision lengths differ significantly in both network types. We let n, the number of CMs, vary between 10 and 150. The size of the contention window depends on the number of transmission failures i for the packet and was equal to:

$$\begin{aligned} W_i &= 2^i CW_{\min} & i &\leq m', \\ W_i &= 2^{m'} CW_{\min} & m' &\leq i \leq m. \end{aligned} \tag{2}$$

The minimum contention window $CW_{\min}$ is increased by a factor of 2, starting from 4 to 16, the maximum number of retransmissions m equals 15 and the maximum value of the contention window $CW_{\max}$ varies between 128 and 512, i.e., $m' = 5$. The simulation is written in the C++ programming language and the duration of the simulation is 10^7 contention slots. Figure 4(a) shows that the analytical model is fairly accurate and both the analytical model and the simulation show the same tendencies. The largest deviations are observed for very small or for large populations. It can be noted that the analytical model performs better as the minimum contention window $CW_{\min}$ grows. Figure 4(a) also depicts that the saturation throughput strongly depends on the number of stations in the network. In particular, the throughput degrades, in most cases, when the network size increases; caused by an increased collision probability. As expected, this decreasing tendency is more pronounced for small values of the minimum contention window $CW_{\min}$. It can also be observed that the analytical saturation throughputs $S(n)$ are optimistic when n is large (say, $n \geq 50$), especially when $CW_{\min}$ is small.

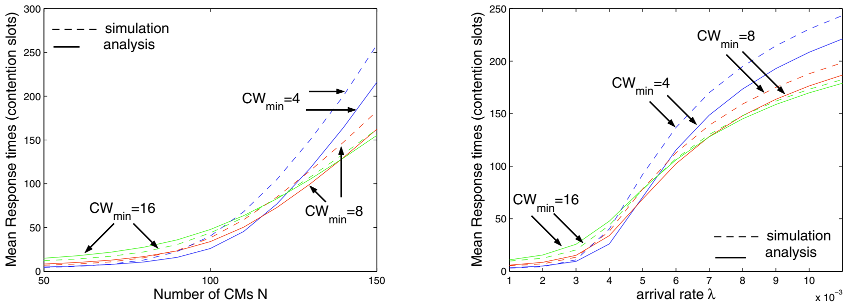
5.2 The Contention Channel

In this section we compare the performance of the contention queue with the performance results obtained by simulating the BEB under Poisson arrivals and



(a) (b)

Fig. 4. (a) Saturation Throughput: analysis versus simulation, (b) Model used to validate the contention queue



(a) (b)

Fig. 5. Validating the contention queue by comparing the mean response time for various parameter settings

a finite population of size N (see Figure 4(b)). The simulation results were gathered after 10^7 contention slots.

Figure 4(b) shows the analytical model containing the contention queue. It is a closed system consisting of 2 service centers and N identical customers. Service center 1 has an infinite server (IS) queueing discipline, while service center 2 is our contention queue. This queueing network is also of the BCMP type and hence, we can rely on the same algorithms as mentioned in subsection 4.1, to compute the mean response time of the contention queue and compare it with the simulation results. We let N, the number of CMs, vary between 50 and 150 and the arrival rate λ is chosen small (between 0.001 and 0.011). The choice of λ was determined by studying the average number of simultaneously contending CMs for different values of N and $CW_{\min}$. According to the Cisco document *"Cisco - Understanding Data Throughput in a DOCSIS World"*, a realistic value for this average number is 10. Finally, we set m and m' equal to 15 and 6, respectively.

Figure 5(a) shows the mean response times as a function of N when λ equals 0.004, whereas in Figure 5(b) λ varies and N is fixed at 100. Both figures indicate that there is a good agreement between simulation results and the analytical model. Similar to what we observed in the previous section, the deviation is somewhat larger for lower values of the minimum contention window $CW_{\min}$. This is understandable as the saturation throughputs $S(n)$ are used as the service rates of the contention queue. The differences between the analytical and simulation results are mainly caused by the inaccuracy of the (optimistic) saturation throughput values $S(n)$.

5.3 The Decomposition Scheme

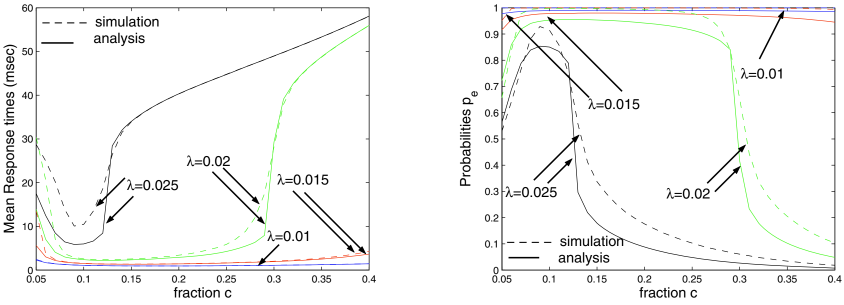
This subsection compares the analytical results obtained by means of the decomposition scheme with a detailed event-driven simulation of the original open queueing network with blocking. The parameter values used in this Section are presented in Table 1. Most of these parameters are chosen as the default values proposed in the Cisco specification *"Cisco - Understanding Data Throughput in a DOCSIS World"*. We assume an average packet length of 438 bytes, this includes an 8% forward error correction (FEC) overhead, the 6 byte overhead of the DOCSIS header, etc. The 10 Mbit channel supports both the contention channel and the reservation channel. The fraction of contention channel, resp. reservation channel, is represented by c, resp. $1-c$, thus the mean service rate of the reservation channel, ν, and the mean service rate of the contention channel, $\mu(n)$, equal:

$$\nu = (1-c)\frac{10^4}{438*8} \tag{3}$$

$$\mu(n) = cS(n)\frac{10^4}{16*8}. \tag{4}$$

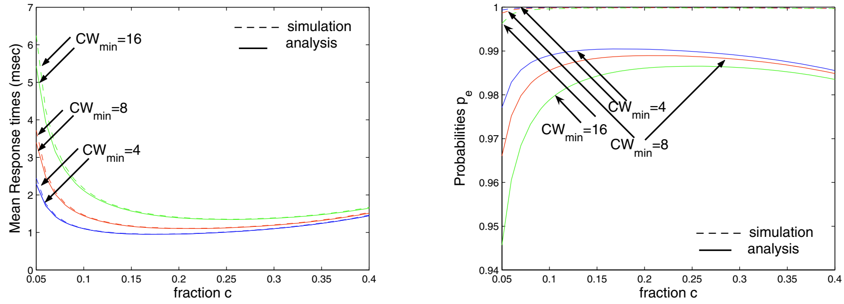
Table 1. Default System Parameters

Parameters	Symbol	Value
Upstream channel capacity	r	10^4 bits/msec
request	Q	16 bytes
data packet	L	438 bytes
arrival rate	λ	0.01 p/msec
number of stations	N	100
size of queue	B	10
min. contention window	$CW_{\min}$	4
max. value	m'	6
max. backoff stage	m	15



(a) (b)

Fig. 6. (a) Validating the decomposition technique: Mean response times, (b) Validating the decomposition technique: The probability of arriving in an empty transmission buffer p_e



(a) (b)

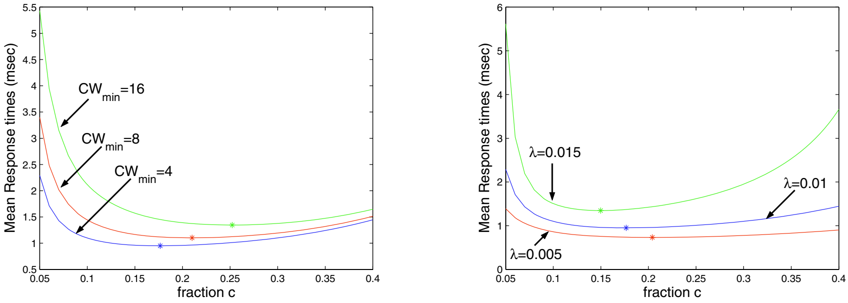
Fig. 7. (a) Validating the decomposition technique: the influence of the minimum contention window $CW_{\min}$ on the mean response time, (b) Validating the decomposition technique: the influence of the minimum contention window $CW_{\min}$ on p_e

Remark that the unit of these rates is $1/msec$ (this is in correspondence with the fact that the arrival rate λ is expressed in packets/msec). The data load of the system can be computed as $\rho_{\text{data}} = \lambda N \frac{438*8}{10^4}$, while the effective load on the reservation channel is obviously higher and equals $\rho_{\text{res}} = \frac{\rho_{\text{data}}}{1-c}$. A comparison of the mean response time and the probability that a new packet finds the transmission buffer empty upon arrival p_e, for different values of the arrival rate λ, is made in Figure 6(a) and Figure 6(b), respectively. Although there is no perfect agreement between the analytical and simulation results, especially for larger arrival rates λ. We observe that the location of the optimal c value, that is, the optimal fraction dedicated to the contention channel, does agree to a high level of accuracy. The sudden increase in the response times are caused by the fact that the effective load on the reservation channel ρ_{res} becomes larger than 1. Similar results were obtained for different N values (e.g., $N = 50, 200$).

The influence of the minimum contention window $CW_{\min}$ is depicted in Figures 7(a) and 7(b). We see that the analytical model is quite accurate: the analytical results for the mean response time nearly coincide with the simulation results. The analytical results for the probability p_e do deviate a little from the simulation results.

6 Numerical Results

In this section we study the optimal fraction c^* of the MAP length that should be dedicated to the contention channel such that the mean response time is minimal. Besides the mean response time, we also evaluate the loss probability p_{loss} and the probability that a new packet finds the transmission buffer empty upon arrival. Unless otherwise stated the protocol parameters are set as indicated in Table 1.



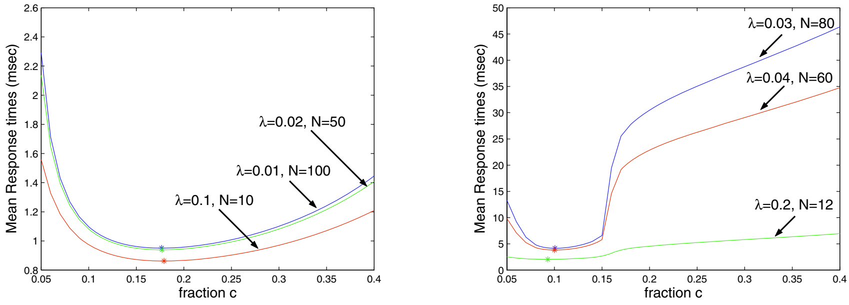
(a)

(b)

Fig. 8. (a) Mean response time when varying $CW_{\min}$, (b) Mean response time when the arrival rate λ varies

We start by investigating the impact of the minimum contention window $CW_{\min}$ on the optimum c^*. When plotting the influence of the contention fraction c we add a "*" to identify the optimum c^*. Figure 8(a) shows that the optimum c^*, as well as the minimum mean response time, increases as a function of $CW_{\min}$. Such an increase reduces the collision probability, because, on average, a station defers its transmission by a greater number of contention slots. Thus, a CM needs less attempts to transmit a request, but the time between two attempts is substantially larger, causing higher delays on the contention channel and augmenting the need for more contention slots.

Experiments not included here, have shown that the transmission buffer size B has no significant influence on the mean response time or on the probability p_e



(a) (b)

Fig. 9. (a) Mean response time for a fixed load $\rho_{\text{data}} = 0.35$, (b) Mean response time for a fixed load $\rho_{\text{data}} = 0.84$

that a packet finds the transmission queue empty (unless it is chosen extremely small), the loss probability obviously does increase for smaller buffers. We have also altered the maximum contention window $CW_{\max}$ from 128 to 512, meaning that m' varies between 5 and 7, but this had no significant impact on the performance of the system either.

Let us now focus on the arrival rate λ. Increasing λ corresponds to a higher data load ρ_{data}. Consequently, the probability that a new packet finds an empty transmission buffer diminishes. Thus, more request packets can make use of the piggybacking scheme and the contention channel becomes less needed. Figure 8(b) confirms that the optimal fraction c^* of the contention channel is lower. Clearly, the minimum mean response time and the loss probability increase as a function of the load ρ_{data}. It is worth noticing that the optimum is quite broad, meaning that even if the fraction c is not exactly equal to c^* we still have a nearly minimum mean response time. The optimum does become somewhat less broad for higher arrival rates λ, which is intuitively clear as the choice of c becomes more critical as more bandwidth is being consumed. As such we recommend to dimension the contention channel for high load traffic scenarios as this will also guarantee a close to optimal performance for the low load cases as well. Similar results are observed when the number of CMs N is varied and λ is held constant.

Finally, we examine the influence of having a fixed load ρ_{data}, but a varying number of CMs N (i.e., λN is kept constant). When the number of CMs N grows, which corresponds to a decrease in the arrival rate λ, the traffic on the data channel becomes less regular, creating a greater need for a larger reservation channel. On the other hand, the probability p_e that a new packet arrives in an empty transmission buffer is substantially higher when the number of CMs N increases (and ρ_{data} is fixed), leading to a greater demand for a larger contention channel as well. The tradeoff between these two opposing forces will decide whether the optimum c^* de- or increases with N. Figure 9(a) shows some combinations for N and λ that result in a data load of 0.35, whereas Figure 9(b) corresponds with a data load of 0.84. Numerical results not included here, have shown that the probability p_e increases much more severely as a function of N for the $\rho_{\text{data}} = 0.84$ scenario. Therefore, we may conclude that for higher loads the need for more contention channel capacity, has a stronger impact, which results in a higher optimal fraction c^*, whereas for lower loads the burstiness of the traffic gets the upper hand, causing a reduction of the fraction c^* assigned to the contention channel.

7 Conclusion

Based on the results presented in section 6 we recommend to dedicate 10 to 15% of the minislots part of the MAP length to the contention channel as this will result in a near optimal mean response time for all arrival rates λ. Indeed, for high data loads the optimum is small and close to 10%, whereas for lower data loads the optimal fraction is larger, but broader as well, meaning that the a fraction between 10 to 15% still provides near optimal results. The exact optimal fraction c^* depends on several system parameters, e.g., the minimum contention window $CW_{\min}$, the number of CMs, etc. In view of the pessimistic nature of the Poisson arrival process when dimensioning the contention channel, assigning 10% of the available uplink channel to the contention channel should be sufficient for real world traffic scenarios.

Acknowledgment

B. Van Houdt is a post-doctoral Fellow of the FWO-Flanders.

References

1. F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the Association for Computing Machinery*, 22(2):248–260, Apr 1975.
2. G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Area in Comm.*, 18(3), Mar 2000.
3. CableLabs. *Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification.* Available at http://www.cablemodem.com/specifications.
4. C. H. Foh and M. Zukerman. Performance analysis of the IEEE 802.11 MAC protocol. In *European Wireless 2002*, Florence, Feb 2002.
5. S.S Lavenberg. *Computer Performance Modeling Handbook.* Academic Press, 1983.
6. R. Litjens, F. Roijers, J.L. van den Berg, R.J. Boucherie, and M.J. Fleuren. Performance analysis of wireless LANs: an integrated packet/flow level approach. In *Proceedings of ITC-18*, Berlin, Sep 2003.
7. S. Ramesh and H. G. Perros. A multilayer client-server queueing network model with synchronous and asynchronous messages. *IEEE transactions on software engineering.*, 26(11), Nov 2000.
8. H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma. Performance of reliable transport protocol over IEEE 802.11 wireless LAN: Analysis and enhancement. In *IEEE Infocom'2002*, New York, Jun 2002.

Appendix A: Modeling the Binary Exponential Backoff Algorithm

Assume a fixed number of contending stations n, each always having a packet available for transmission. In part one of the analysis the behaviour of a single station is modeled by a Markovian model. The objective of this part is to obtain the stationary probability τ that a station transmits a packet in a randomly chosen slot. The second part expresses the throughput $S(n)$ as a function of τ.

Part 1. The key approximation made by Bianchi [2] is that, at each transmission attempt and regardless of the number of retransmissions suffered, each packet collides with a constant and independent probability p, called the conditional collision probability. Once the independence is assumed, two stochastic variables can be used to model the protocol behavior (see in Figure 10). Let $b(t)$ be the random variable representing the remaining backoff interval size for the given station at slot time t and let $s(t)$ represent the backoff counter value $(0, \ldots, m)$ of the tagged station at time t. The bi-dimensional process $\{s(t), b(t)\}$ is then a discrete-time Markov Chain (MC). The stationary distribution of this MC, denoted by $b_{i,k}$, for $0 \leq i \leq m$ and $0 \leq k \leq W_i - 1$, is given by Eqn. (5), see [8].

$$
\begin{aligned}
b_{0,0} &= \begin{cases} \frac{2(1-2p)(1-p)}{CW_{\min}(1-(2p)^{m+1})(1-p)+(1-2p)(1-p^{m+1})} & m \leq m' \\ \frac{2(1-2p)(1-p)}{CW_{\min}(1-(2p)^{m'+1})(1-p)+(1-2p)(1-p^{m+1})+CW_{\min}2^{m'}p^{m'+1}(1-2p)(1-p^{m-m'})} & m > m' \end{cases} \\
b_{i,0} &= p^i b_{0,0} \qquad \text{for } 0 \leq i \leq m \\
b_{i,k} &= \frac{W_i - k}{W_i} b_{i,0}, \qquad \text{for } 0 \leq i \leq m
\end{aligned}
\tag{5}
$$

The probability τ depends on the conditional collision probability p and can be expressed as

$$
\begin{cases} \tau = \frac{1-p^{m+1}}{1-p} b_{0,0} \\ p = 1 - (1-\tau)^{n-1} \end{cases} . \tag{6}
$$

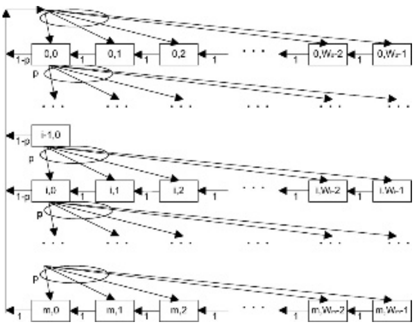


Fig. 10. Markov chain model of new backoff window scheme

This nonlinear system of equations in the unknowns τ en p can be solved easily by numerical techniques.

Part 2. In the second part the events that can occur within a randomly chosen contention slot are studied in order to get a formula for the system throughput $S(n)$, defined as the probability that a successful transmission occurs in an arbitrary contention slot when n stations contend on the channel. Since n stations contend on the channel and each transmits with probability τ, we can define the probability P_{tr} that there is at least one transmission in the considered slot time as $P_{\mathrm{tr}} = 1 - (1 - \tau)^n$, and the probability P_{s} that a transmission occurring on the channel is successful is given by $P_{\mathrm{s}} = \frac{n\tau(1-\tau)^{n-1}}{P_{\mathrm{tr}}}$. We can conclude that the saturation throughput $S(n)$, which equals the probability that a successful transmission occurs in an arbitrary slot, equals $P_{\mathrm{s}}P_{\mathrm{tr}}$, thus $S(n)$ becomes $S(n) = n\tau(1 - \tau)^{n-1}$. Notice, this formula is a simplified version of Bianchi's as the request packets, empty slots and collision periods all have a fixed length equal to one contention slot.

CONTI: Constant-Time Contention Resolution for WLAN Access

Zakhia G. Abichar and J. Morris Chang

Dept. of Electrical and Computer Engineering,
Iowa State University, Ames, IA 50011, USA
{abicharz, morris}@iastate.edu

Abstract. Designing an efficient and fair access control protocol is a challenging task in the field of wireless networks. Often in the known schemes, one of the important performance metrics is enhanced at the expense of another. In this paper, we present a distributed access scheme that is based on the binary countdown mechanism. The proposed scheme has two main features: 1) the ability to resolve the contention in a constant number of time slots, hence the name constant-time (CONTI) and 2) a very low collision rate even at large network sizes. The simulation results show that CONTI outperforms the IEEE 802.11 DCF scheme in all the essential performance metrics: CONTI achieves a higher throughput by up to 55%, reduces the collision rate by up to 84%, renders the delay less variant and exhibits high fairness.

1 Introduction

The area of wireless access protocols has been an active research arena in the last decade. The main goal of shared medium access protocols is to achieve high efficiency and fairness under various network configurations. The wireless protocol standard that is dominantly in use today is the IEEE 802.11. The standard describes two modes of operation: the distributed coordination function (DCF) and the point coordination function (PCF). DCF is a distributed scheme that is designed to support best-effort traffic. On the other hand, PCF is a centralized scheme where stations are polled to ensure their service requirements are met.

In the IEEE 802.11e draft [1], which is proposed for quality-of-service (QoS) support, the DCF is enhanced through a scheme where each station runs up to four instances of DCF; this scheme is known as enhanced distributed channel access (EDCA). In addition, the PCF scheme is enhanced through a scheme where the access point is given increased priority to allow a better service for QoS traffic; this scheme is known as the hybrid coordination function (HCF). All of the aforementioned schemes are built on top of DCF, which increases the importance of the basic scheme.

Performance evaluation has shown that DCF performs reasonably well for transmitting best-effort packets in small-size networks. Its medium utilization has been shown to approach 68% in [2]. However, DCF leads to significant throughput degradation for networks of large size [2]-[3]. The main reason for the throughput drop under DCF is contributed to the high collision rate which reaches 40% (presented in section 6). Making use of this observation, we develop a scheme that is based on an efficient collision

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 358–369, 2005.

resolution mechanism called the binary countdown mechanism [4]. The main goal of the scheme is to allow for a low collision rate even when the network size grows.

Before a channel access occurs, CONTI eliminates a portion of the contending stations after the elapse of one contention slot. A powerful elimination mechanism is presented which is capable of reducing the number of stations from 100 stations to 5 stations in just one contention time slot. This elimination is achieved based on a randomly chosen value of a local Boolean variable. The value of the Boolean variable is then refreshed and the process is repeated several times until one station remains in the set of contending stations. As a result, the number of time slots that precedes a channel access is of logarithmic complexity.

According to CONTI, the collision rate is directly dependent on the number of contention slots: the longer the contention period, the lower the collision rate. Thus, a low collision rate can be achieved independently of the number of contending stations. Moreover, the resulting scheme exhibits a desired behavior: for a wide range of scenarios, CONTI can be efficiently run with the same number of contention slots. This would make the channel access time constant while keeping the collision rate minimal. Finally, the simulation studies show that CONTI outperforms DCF in all the essential performance metrics: throughput, delay, fairness and rate of collisions.

The rest of this paper is organized as follows. Section 2 discusses related work and section 3 briefly describes the operation of the IEEE 802.11 DCF. The proposed scheme is presented in section 4 and a mathematical analysis to determine the parameters of CONTI is presented in section 5. The simulation results are presented in section 6 and, finally, the paper is concluded by summarizing remarks in section 7.

2 Related Work

A brief history on the milestones in wireless networking protocols is included in the complete form of this paper, publicly available as a technical report [5]. This part has been omitted from this paper due to space limitation. We limit the discussion in this section to schemes that are closely related to CONTI.

The scheme we propose is based on a mechanism that is efficient in collision resolution called the binary countdown mechanism. In this section we give a description of the binary countdown mechanism and clarify the contribution of our work. In addition, we elucidate the difference between the binary countdown mechanism and a class of access protocols known as binary tree protocols. The idea behind the binary tree protocols is close to that of the binary countdown mechanism. However, there are significant hurdles in applying binary tree protocols to wireless networks [6]. On the other hand, the binary countdown mechanism can be easily applied to wireless networks.

The binary countdown mechanism has been described in [4] for wired networks access. However, this mechanism had a major problem in the form it was presented which made limited its practical use. To access the channel, stations run through a contention period whose length (in time slots) is equal to the number of bits representing the stations' addresses. Each time slot corresponds to a bit position, with the earliest time slot corresponding to the most significant bit. During each time slot, a station that has a value of one in the corresponding address bit jams the medium. Stations having a value of zero for that bit sense the medium. If the medium has been jammed, the listening stations retire from the current contention. Through this procedure, the station with the highest address is guaranteed to win the contention.

The main limitation in the binary countdown mechanism is letting the station with the highest address win. On the other hand, the main advantage of the binary countdown mechanism is collision-free communication since addresses are unique identifiers. In this paper, we re-discover the mechanism of binary countdown and consider the key parameters that allow to obtain the best of this idea. Our main contribution lies in addressing the two critical parameters that control the operation of the binary countdown mechanism. Those parameters have not been mathematically analyzed before:

- The binary countdown session should be run for a certain number of time slots. The number of time slots should be kept minimal so as not to waste time. On the other hand, the contention time should be high enough to allow for an efficient contention resolution procedure. We develop the necessary mathematical analysis in section 5 to address this issue.
- For each time slot a station needs to decide whether to jam or sense the medium. Making this decision based on the address of the stations is not at an optimal solution. We present a mechanism that chooses those bits in such a way to quickly reduce the number of contending stations. The resultant mechanism is remarkably powerful and allows reducing the number of stations from 100 to 5 in just one time slots.

In the recent literature, the binary countdown mechanism has been employed to underlie access schemes for wireless networks [7]-[8]. However, the mechanism was used in a sub-optimal manner. The probability of selecting bit values was chosen purely randomly [8]. In addition, the number of contention slots was not kept at a minimum and many components have been added to it to maintain fairness.

A class of algorithms that is close to the binary countdown mechanism has been proposed in the 1970s by Tsybakov [9], Capetanakis [10] and others. Those algorithms are known as binary tree protocols. In the binary tree schemes, stations are allowed to collide and then colliding stations are divided into groups. Then, the groups are processed in a certain order and collisions within these groups are resolved by recursively dividing the concerned groups until a group contains one stations.

The shared aspect between binary tree protocols and the binary countdown mechanism is that, in both cases, the stations are divided into groups. However, binary tree protocols allow collisions to occur since, in wired networks, collisions can be detected and an ongoing transmission is halted, if need be. In wireless networks, it is not possible to detect collisions while transmitting.

A scheme has been proposed [11] to apply binary tree protocols to wireless networks. The scheme uses a separate feedback channel that informs the stations whether the transmission has collided or not. The feedback is provided by an access point. This solution, however, has few limitations: 1) there is a need for a separate channel, 2) even though the stations are informed of the results of a collision, the feedback is received after the whole frame is transmitted and 3) there is a need for an access point which renders the scheme inapplicable to ad-hoc networks. On the other hand, the proposed scheme CONTI is free of those limitations.

3 Operation of the IEEE 802.11 DCF

The contention-based access mechanism that is adopted in the 802.11 standard is the DCF. A successful frame transmission cycle under DCF consists of the transmission of a data frame and the reception of the corresponding ACK frame by the source station (Fig.1). In addition, a transmission cycle may include one or more collisions.

Before transmitting a data frame, a station senses the medium. If the medium is idle, the station initiates the transmission. Otherwise, the station proceeds to follow a two-step procedure. First, the station waits until the medium becomes idle and then defers for an interframe space (IFS) that is defined for each class of frames (DIFS for data frames). If the medium remains idle throughout the IFS, the station proceeds to the second step which consists of setting a backoff timer to a value that is uniformly chosen from the contention window (CW) interval. The purpose of the second step is to reduce the chance of a collision between stations that are transmitting data frames. After that, the station reduces its backoff timer by one unit (slot time) following the elapse of an idle time slot. If the medium ceases to be idle during any of the time slots, the station pauses its backoff timer and resumes it after detecting the medium to be idle for another IFS time frame.

In the event where the backoff timer reaches zero, the station transmits its packet. Should a collision occur, the involved stations defer and each of those stations doubles its CW if it were less than the defined maximum CW size. In the case of a successful packet transmission, the destination station defers for a short IFS (SIFS) and then sends back an ACK packet. Moreover, the transmitting station resets its contention window to the minimum defined value (CW_{min}). More elaborate details on the operation of the DCF scheme can be found in the IEEE 802.11 standard document [12].

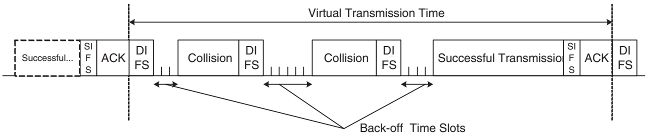


Fig. 1. DCF transmission cycle

4 Proposed Scheme

To provide contention access to the medium, CONTI runs a certain number of contention slots to resolve the contention before a transmission can be initiated. This period will be referred to as the contention resolution period (CRP) henceforth (Fig.2). During each contention slot, a portion of the contending stations is eliminated. When a station is eliminated, it is meant that it will cease contention to the medium during the current CRP. Thus, it can contend again after the occurrence of one transmission trial.

The elimination of stations is achieved through the use of a local Boolean variable called the $try-bit$. Each station randomly chooses a value for its try-bit and either jams ($try - bit = 1$) or senses the medium accordingly. A station retires only if its try-bit is equal to zero and the medium was jammed. The stations with try-bit equal to one do not retire at this contention slot. After the elapse of one time slot, the stations that were not eliminated refresh their try-bit variables and repeat the same process. This process is repeated until one station remains in the set of contending stations. Therefore, this station will initiate a transmission.

This procedure proves to be capable of quickly reducing the number of contending stations if the relevant parameters are carefully chosen. The two relevant parameters are: 1) the number of time slots during which the contention resolution runs and 2) the probability of jamming or sensing the medium during a given time slot. Those two issues are addressed in section 5.

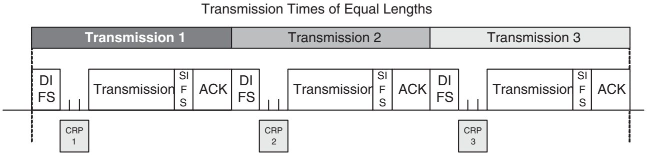


Fig. 2. Operation of CONTI

In general, the length of the CRP should increase with an increase in the number of operating stations. However, we show in the mathematical analysis of section 5 that the collision resolution mechanism is flexible enough to let CONTI run with a constant number of contention slots for a wide range of scenarios.

We finally mention the following notes: during a contention slot, if all the stations choose a value of one for the try-bit, then all of them would jam the medium and thus move to the subsequent contention slot. On the other hand, if all the stations choose a value of zero for the try-bit, the medium remains idle and, again, all the stations proceed through the CRP.

It is noteworthy to mention that it is not possible for all the stations to retire from the contention. This is because a station retires, only if there is another one who jammed the medium. Thus, the station that has jammed the medium does not retire.

The proposed scheme does not totally eliminate the chance of the occurrence of a collision. In the event where two (or more) stations choose the same value for their try-bits in the last contention time slot, those stations will collide.

5 Mathematical Analysis

We present in this section a mathematical analysis that allows for choosing the two parameters for CONTI: the number of contention time slots and the probability for assigning a value of one for the try-bit during a given time slot.

5.1 Choosing a Value for the *try-bit* with Probability p

In the process of contention, a station chooses a value for its try-bit variable during each time slot. The chosen value is equal to one with probability p. This value determines whether the station will jam or sense the medium.

We highlight the following two factors. *Factor A:* If the value of p is high, most of the stations would choose a try-bit of value one and thus move to the next contention slot. In this case, the collision will be resolved in a slow manner. On the other hand *(Factor B)*, an extremely low value of p is not desirable. In that case, all stations may get a value of zero and thus no stations would be eliminated. This is especially true for a low number of contending stations, which occurs during the last couple of the contention slots.

At the early time slots of the contention, the number of competing stations is large and is reduced as the contention proceeds. Thus, the value of p is made small at the early contention slots *(due to Factor A)* and then elevated as the contention proceeds *(due to Factor B)*. To avoid elevating the value of p in the last time slot more than needed, we make use of the *Lemma* that is presented later in this section, where the optimal value of p is found for the last contention slot.

After the elapse of a contention slot, the probability that i stations, out of n contending stations move to the subsequent contention slot is given by the following equation:

$$pr(i) = \begin{cases} \binom{n}{i}.p^i.(1-p)^{n-i} & 1 \leq i \leq n-1, \\ p^n + (1-p)^n & i = n. \end{cases} \tag{1}$$

It follows that the expected number of stations, E, that will move to the subsequent contention slot is given by the following equation:

$$E = \left(\sum_{i=1}^{n} i.\binom{n}{i}.p^i.(1-p)^{n-i} \right) + n.(1-p)^n. \tag{2}$$

Solving this equation numerically, we can determine the optimal values of p that allow for the fastest reduction in the number of stations between two consecutive contention slots.

For a given number of stations, there is an interval for the values of p that allows for the steepest reduction in the number of stations per time slot. Fig.3 plots those intervals as the number of stations varies up to 100. The digit that is placed over the interval lines represents the minimum number of stations that could remain in contention after the elapse of one slot time. For example, starting with 20 stations in a given contention slot, a value of p in the interval [0.12 – 0.18] reduces the number of stations to an expected value of 3, the steepest possible reduction.

Fig.3 also shows a desirable property of this mechanism. We can see that the intervals for several network sizes share many points in common. The dotted line has been drawn to pass through those intervals in a smooth way. It could be noticed that the dotted line is not far from being a horizontal line. This fact allows us to choose a value of p that is close to the optimal value even if the network size is not known. Thus, it makes it feasible to run CONTI under the same tuning for a wide range of scenarios.

Fig.4 shows how fast the number of competing stations can be reduced when the optimal values of p (based on Fig.3) are employed, thus using a minimum number of contention slots. From the data underlying this figure, it takes two contention slots before the number of expected stations is reduced to one when starting from 20 stations or less. As for the other cases, 20 to 100 stations, it takes three contention slots. In

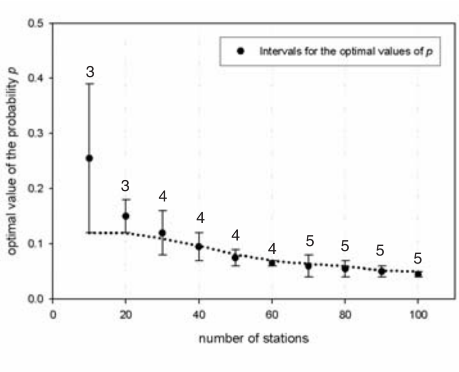


Fig. 3. Intervals of p for the fastest reduction

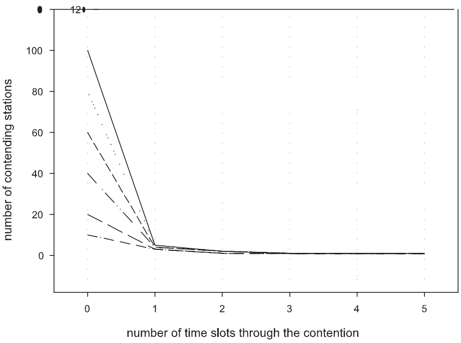


Fig. 4. Fastest contention resolution

practice, it is beneficial to run the contention for one or two additional slots if we need to ensure a low collision rate.

At the last contention slot, a successful transmission occurs in one of the following two cases. The first case: only one station has remained from the previous contention slot. This station will successfully transmit independently of the chosen value for its try-bit. The second case: there is more than one remaining station and the try-bits are chosen in such a way that only one station has a value of one for the try-bit. For the latter case, the optimal value of p is given by *Lemma 1.*

Lemma 1: Given n remaining stations at the last contention slot, the optimal solution demands that each station chooses a value of one for its try-bit with probability $p = 1/n$.

Proof: At the last stage of the contention, a successful transmission occurs if only one station chooses a value of one for its try-bit. It follows that the probability of a successful transmission is given by:

$$p_s = n.p.(1-p)^{n-1}. \tag{3}$$

The value of p that maximizes p_s is obtained by solving the following equation (4) which leads to $p = 1/n$.

$$\frac{dp_s}{dp} = n.(1-p)^{n-1} - n.p.(n-1).(1-p)^{n-2} = 0. \tag{4}$$

■

In practice, the value of p will be set to $1/2$ in the last contention slot. This is because the number of expected stations that remain in the last contention slot is equal to two, based on the information from Fig.4.

6 Performance Evaluation

This section presents the simulation studies that were carried to evaluate CONTI and compare its performance to the IEEE 802.11 DCF. We developed a packet-level simulator to conduct the simulation studies. The IEEE 802.11 standard was run using the DSSS specifications. The parameters used for the simulation are summarized in Table 1. It was assumed that best-effort data packets are available at stations at all times. Separate simulation runs were carried for various packet sizes ranging from 50 bytes to 1250 bytes at 2 Mbps. In addition, simulation studies were conducted at the rate of 11 Mbps for packet sizes ranging from 250 bytes to 2346 bytes. The results for 11 Mbps were omitted due to space limitation and due to the fact that they exhibit similar trends to those shown at 2 Mbps. Those results are included in the technical report version of this paper [5].

For the presented simulation results, CONTI was run for six time slots. This number of the time slots has resulted in the best values of the throughput when jointly considering all the presented scenarios. The values of p that were used during each of the six time slots are shown in Table 1.

Table 1. Simulation Parameters

Parameters for DCF and CONTI	Parameters for CONTI
DIFS duration: $50\mu s$	#contention slots: $k = 6$
SIFS duration: $10\mu s$	$p_i : i = [1 : 6]$
Slot time: $20\mu s$	$[0.07, 0.2, 0.25$
Channel rate: $2Mbps$	$0.33, 0.4, 0.5]$

Figures 5, 6 and 7 show a throughput comparison between CONTI and DCF when the channel rate is 2 Mbps and the network size is 10, 50 and 100 stations, respectively. The measured normalized throughput is defined as the data rate (including packet headers) divided by the channel rate. In addition, the ideal throughput of CONTI is represented by the dotted line. The ideal throughput can be attained if the number of competing stations is known to all the stations. The packet sizes for those simulation runs vary from 50 bytes to 1250 bytes.

For a network size of 10 stations, CONTI attains a throughput of 92.4% while DCF reaches a throughput value of 82.2%. However, as the the network size grows larger, the disparity in performance between CONTI and the DCF is widened. For 50 stations, CONTI reaches a throughput of 91.5% while the standard's throughput attains 66.5%. Finally, for the case of 100 operating stations, CONTI achieve a throughput of 90.4% while the standard's throughput is 58.5%.

In the case where the network size is 10 nodes (Fig.5), DCF allows for a better performance than CONTI for packet sizes that are less than 75 bytes. The reason for this trend is twofold. Firstly, at a low network size, the collision rate of DCF is low even though it is still four time higher than that of CONTI (measurements of collision rates are presented in Fig.8). Thus, the low collision rate of CONTI is not much advantageous over DCF. Secondly, the average access time of DCF for a 10-node network is less than six time slots (as measured: about 3 slots). Then, the combination of those two factors allows DCF a better performance for the considered specific scenario. However, as the packet size grows larger, CONTI outperforms DCF. This is because the cost of a collision increases for large packet sizes. We note that the chances of transmitting a packet of size smaller than 75 bytes at 2 Mbps is very small. Thus CONTI retains the advantage in the majority of cases.

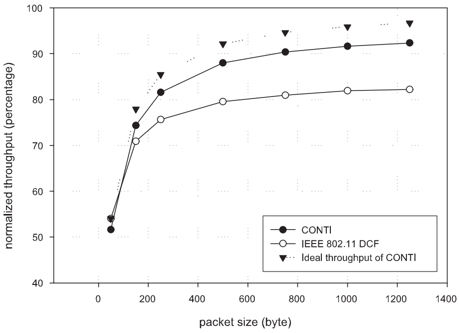


Fig. 5. Throughput for 10 stations at 2 Mbps

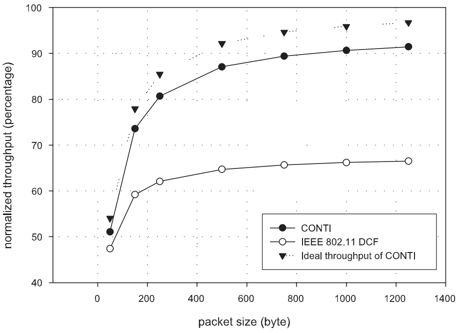


Fig. 6. Throughput for 50 stations at 2 Mbps

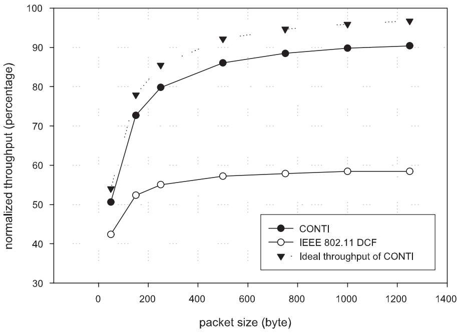


Fig. 7. Throughput for 100 stations at 2 Mbps

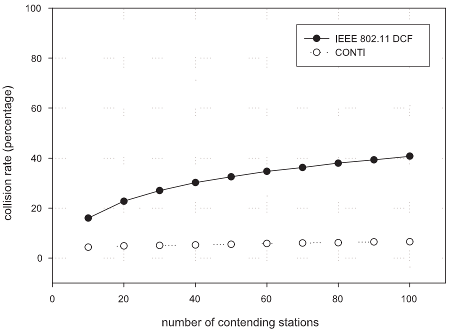


Fig. 8. Collision rates

As we mentioned in the introductory section, the main motivation of CONTI was to allow for a low collision rate. The measurement of this factor is presented in Fig.8. This figure shows the collision rates for CONTI and DCF as the number of contending stations varies from 10 to 100 stations. The packet sizes in this case are set to 1000 bytes. It could be noticed that there is a great disparity in the collision rates between the two schemes. For the shown cases, the collision rate of CONTI varies from 4.37% to 6.37% while that of DCF varies from 16.00% to 40.75%. This observation is significant in giving insight about why CONTI outperforms DCF: the high collision rate of DCF impedes other essential performance metrics such as the throughput, delay and fairness.

Figures 9 and 10 show a comparison of the delay distribution when the network consists of 10 and 100 stations, respectively. The delay metric that is presented in the following plots is measured from the moment the packet reaches the head-of-line in the queue to the instant it is successfully received at the other party. For the following simulation runs, the packet sizes were set to 1000 bytes.

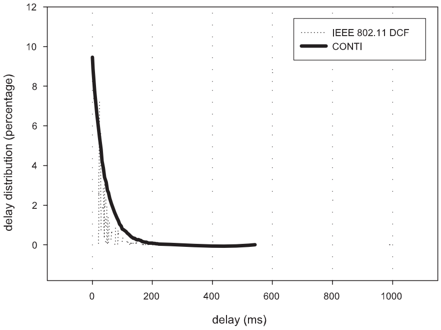


Fig. 9. Delay distribution for 10 stations

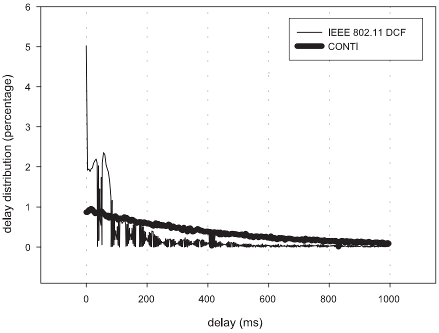


Fig. 10. Delay distribution for 100 stations

In Fig.9, the delay distribution of CONTI is represented by the thick line. The worst case delay for CONTI in this case is 542 ms while that of DCF is 5.07 seconds. It could also be noticed that the delays of the packets that correspond to DCF are much fluctuating. This fluctuation is attributed to the high collision rate of DCF. Since the instant in which a collision can occur is highly random, this renders the delay that is experienced by a packet hard to anticipate. However, for the case of CONTI the delay distribution is much smoother.

For the case of 100 operating stations, the worst case delay for CONTI is 5.2 seconds whereas the corresponding one for DCF is 25.93 seconds. However, from the data underlying Fig.10 (also shown in the figure), it could be noted that about 5.02% of the packets under DCF are transmitted with an infinitesimal delay while only 0.87% of the packets under CONTI are transmitted with that minimal delay, which is close to zero. This trend is attributed to the following reason: the average access time of DCF is smaller than that of CONTI. Thus, in a given time interval, more transmission trials are initiated under DCF. This gives the chance of having more occurrences of the infinitesimal delay case. However, due to the high collision rate of DCF, a fairly large delay can occur at any time.

As an overall result, DCF allows for more frequencies of the infinitesimal delay at the expense of many fluctuations in its delay distribution. On the other hand, CONTI favors regularity in the expected delay at the expense that its minimal delay may be higher than that of DCF. This observation renders CONTI a more suitable scheme to be extended for support of real-time traffic where delay anticipation is a precious asset. We also note that the worst-case delay of DCF is much larger than the corresponding one of CONTI.

Figures 11 and 12 present the measurement of the fairness metric for DCF and CONTI. For those simulation runs, the packet sizes were set to 1000 bytes. The simulation was run until 100,000 transmission trials have occurred.

The plots in the figures show the percentage amount of the fair share that each of the contending stations has attained. The fair share is obtained by dividing the number of successfully transmitted packets by the number of operating stations.

For the case of 10 operating stations, the fair share values under CONTI range from 98.34% to 101.31% whereas the counterpart values that correspond to DCF range from 95.00% to 105.83%. At first sight this might seem a bit unexpected since the DCF scheme is well credited in regard to the fairness metric. Taking a closer look at the binary exponential backoff algorithm, it could be realized that a station doubles its contention window size every time it undergoes a collision, thus imposing a penalty on colliding stations. In the short following time range, the colliding stations are disadvantaged with respect to other stations. On another count, all stations are equally likely to collide in the long-run. We then deduce that the time range that takes DCF to be fair is greater than the corresponding range of CONTI. This could be interpreted by the fact that CONTI is a memoryless protocol.

The same trend that was described in the previous paragraph is also exhibited for the case of 50 contending stations. In this scenario, the fair share values under CONTI range from 96.88% to 105.67% whereas the corresponding values to DCF range from 77.81% to 116.67%. Finally, we mention the following observation: the fair share values at 50 stations exhibit a wider deviation than those measured at 10 stations. This is due to two reasons: 1) for an undiscriminating scheme, the longer the simulation time is, the closer the share values are to 100% and 2) the number of transmission trials has remained the same for both of the simulation scenarios.

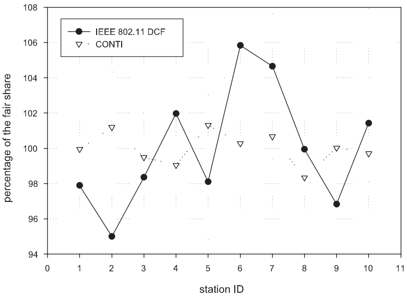


Fig. 11. Fairness measure for 10 stations

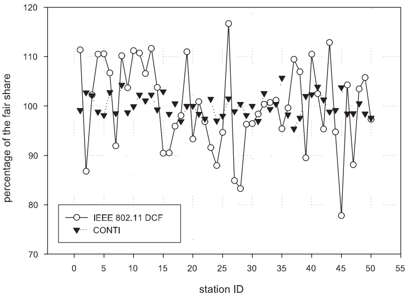


Fig. 12. Fairness measure for 50 stations

7 Conclusion

In this paper, we propose a contention-based MAC scheme that is based on the binary countdown mechanism. In CONTI, an efficient mechanism is employed to allow for a fast contention resolution and a low collision rate. We also present a mathematical analysis that allows to choosing proper values for the parameters of CONTI. Finally, the presented simulation studies show that CONTI outperforms DCF in all of the essential performance metrics: 1) CONTI allows for a higher throughput for the vast majority of the scenarios. 2) A given value of the fairness metric could be achieved in a shorter time range over CONTI as compared to DCF. 3) The delay distribution under CONTI is much smoother than that of DCF and the worst-case delay of CONTI is much smaller. 4) The collision rate of CONTI is, by a large margin, smaller than that of DCF. In summary, CONTI proves to be an efficient wireless MAC scheme that is capable of keeping a balance between all the essential performance metrics.

References

1. IEEE 802.11 WG, Draft Supplement to Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), IEEE 802.11e/Draft 8.0, February 2004.
2. Bianchi, G.: Performance Analysis of the IEEE 802.11 Distributed Coordination Function. IEEE Journal on Selected Areas in Communications, Vol.18, No.3, pp.353-547, March 2000.
3. Golestani, S.J: A Self-Clocked Fair Queuing for Broadband Applications. IEEE INFOCOM, 1994, pp.636-646.
4. Tanenbaum, A.S: Computer Networks, 4th edn. Prentice-Hall, Redwood City, CA (2003)
5. Abichar, Z., Chang, M.: CCR: A novel MAC scheme with constant-time contention resolution for WLAN. Iowa State University Technical Report. TR-2004-07-0. August 2004.
6. Bertsekas, D., Gallager, R.: Data Networks, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ (1992)
7. Yeh, C.H.: Distributed Collision-Free/Collision-Controlled MAC Protocols for Mobile Ad Hoc Networks with Hidden Terminals. International Conference on Information Networking(ICOIN), August 2004.
8. Wu H., Utgikar A. Tzeng, N.F.: SYN-MAC: A Distributed Medium Access Control Proto-col for Synchronized Wireless Networks, ACM Mobile Networks, to appear.
9. Tsybakov, B.S., Beloyarov, A.N.: Random multiple access in binary feedback channel. Problemy Peredachi Informatsii. Vol26 (4). October-December (1990) 83-98.
10. Capetanakis, J.I.: Tree algorithms for packet broadcasting channel. IEEE Transactions on Information Theory. Vol25. September (1979) 505-515.
11. Garcs, R., Garcia-Luna-Aceves, J.J.: Collision avoidance and resolution multiple access for multichannel wireless networks. IEEE INFOCOM, 2000.
12. IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11, 1997.

Efficient 3G/WLAN Interworking Techniques for Seamless Roaming Services with Location-Aware Authentication*

Minsoo Lee[1], Gwanyeon Kim[1], Sehyun Park[1]**, Sungik Jun[2], Jaehoon Nah[2], and Ohyoung Song[1]

[1] School of Electrical and Electronics Engineering, Chung-Ang University,
221, Heukseok-dong, Dongjak-gu, Seoul 156-756, Korea
{lemins, cityhero}@wm.cau.ac.kr,{shpark, song}@cau.ac.kr
[2] Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
{sijun, jhnah}@etri.re.kr

Abstract. This paper proposes novel concepts and architectures for location-aware seamless authentication and roaming in the new interworking system between third-generation (3G) mobile networks and wireless local area networks (WLANs) where local mobility movements (micro-mobility) are handled together with global movements (macro-mobility). We introduce location as a key context in secure roaming mechanism for context-aware interworking. The fast secure roaming with location-aware authentication is implemented at an entity called *LBS Broker* that utilizes the concepts of direction of user and *pre-warming zone.* We present the interworking techniques with *LBS Broker* for seamless secure WLAN/3G integration enabling to meet the requirements of the future location-aware service scenarios. Performance evaluation is also presented to demonstrate the effectiveness of the proposed scheme for fast location-aware secure roaming.

Keywords: ubiquitous computing, location-aware, 3G, 4G, WLAN, interworking, security

1 Introduction

In the fourth-generation (4G) mobile networks which are expected to be very complex systems interconnecting various technologies and architectures, new intelligent services will need to be aware of *location* that can determine which types

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC(Home Network Research Center)-ITRC support program supervised by the IITA(Institute of Information Technology Assessment).

** The corresponding author.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 370–381, 2005.

of devices are available and how communication should be conducted. *Location-aware computing* may soon become a part of everyday life with Location-based services (LBS) like asset tracking, environmental resource discovery and control, and electronic tourist guides [1].

Location-aware computing is made possible by the convergence of three distinct technical capabilities: location sensing, wireless communication, and mobile computing systems. As multiple access technologies are becoming part of a common wireless infrastructure, mobility management is more complicated in the integrated networks of wireless LANs (WLANs) with third-generation (3G) mobile networks such as Global System for Mobile Communications/General Packet Radio Service (GSM/GPRS), Universal Mobile Telecommunications System (UMTS) and CDMA2000.

Recently, complementary features between WLANs with high data rates and 3G mobile networks with wide coverage have spurred the demand for 3G/WLAN interworking systems. In these interconnected wireless networks we need smart techniques for determining the current location of a mobile node (MN). Apart from technical difficulties in providing accurate location information, there is also a lack of a clearly defined framework to create innovative security services with location information. There are few safeguards on location privacy and security as in our previous work [2].

On the 3G/WLAN interworking, a feasibility study [3] was conducted by the 3rd Generation Partnership Project (3GPP). However the study does not deal with efficient mobility and security management techniques that are indispensable features for the next generation wireless networks. Based on [4,5,6], we identified location-aware security service requirements and functionalities in 3G/WLAN interworking in Table 1. Scenario 3 allows a customer to access 3G packet-switched (PS) services over WLAN. Scenario 4 allows a customer to change access between 3G and WLAN networks during a service session. Scenario 3 features should be essential for the first stage deployment.

In our paper we focus mainly on scenarios 3 and 4, and propose location-aware secure interworking techniques and architectures that could meet their respective requirements. We discuss how security can be substantially improved through a new form of authentication based on the location-aware security architecture. In line with Denning [6, 7] we suggest location-aware authentication introducing *location* as a new element in a user authentication mechanism. In particular, we propose and discuss efficient mobility management schemes with *LBS Broker* that can support consistent secure LBS provisioning. LBS Broker includes location-aware authenticator for fast secure roaming using the concepts of the direction of the user and pre-warming zone. We further explain how the proposed approach can be applied to seamless secure interworking between 3G and WLAN with the evaluation of our testbed.

The rest of this paper is organized as follow. Section 2 gives related works about location-aware computing in 4G mobile networks. Section 3 identifies the problems and requirements of location-aware security in 3G/WLAN interworking. In Section 4 we propose LBS Broker with location-aware authentication and

Table 1. 3G/WLAN Interworking Requirements with Location-aware Services

Scenarios	Characteristics	Requirements	Related Functions
1 (Loose Coupling)	Common billing and customer care	- Common billing	- Network discovery and selection - Common billing functions
2 (Loose Coupling)	3G-based access control 3G-based access charging	- AAA for 3G subscribers in a WLAN - IP connectivity via WLAN for 3G subscribers - Multimode 3G/WLAN MN	- Network selection with Network Address Identifier(NAI) - AAA Proxy - RADIUS-Diameter interworking function - Authentication with EAP-AKA and EAP-SIM
3 (Loose Coupling)	Access to 3G PS-based services	- User data traffic needs to be routed to the 3G home or visited PLMN - Location Based Service (LBS) via 3G and WLAN	- User data traffic management with Packet Data Gateway (PDG) and Wireless Access Gateway(WAG) - LBS platform with LBS Broker to enforce Location-aware Services
4 (Tight Coupling)	Access to 3G PS-based services with service continuity	- Service continuity for transitions between 3GPP Systems and WLANs. - Changes of QoS - To allow transition of multiple sessions and services	- Location-aware Vertical handover - Location-aware Resource Management - Policy-based location management to enforce location privacy - LBS using Web Services through Internet - LBS service continuity
5 (Very Tight Coupling)	Access to 3G PS-based services with seamless service continuity	- Seamless changes of service - Service change shall not be noticeable to the user	- Service Continuity with fast vertical handover - LBS QoS guarantees - Secure LBS Platform with LBS Broker
6 (Very Tight Coupling)	Access to 3G CS-based services with seamless mobility	- Seamless roaming for 3G PS-based service - CS-based service with WLAN	- Seamless Service Continuity - Transparent roaming - LBS QoS guarantees - Secure LBS Platform for heterogeneous networks

pre-warming for fast secure roaming. Section 5 suggests our location-aware security architecture for 3G/WLAN interworking systems. In Section 6, we discuss experimental result and we conclude in Section 7.

2 Related Works

4G mobile networks are expected to be very complex systems interconnecting various technologies and architectures. On these complex systems new intelligent services will need to be aware of *location* that can determine which types of devices are available and how communication should be conducted in order to maintain QoS guarantees. Location information will be available from various types of network including sensor networks, WLANs, 3G including UMTS and CDMA 2000[8]. According to development of WLANs, 3G and 4G networks and increase of the accuracy of the positioning technology, many LBSs with efficiency and reliability will appear.

Toward seamless security of the interconnected networks, fundamental features such as smooth roaming and interworking techniques, QoS guarantee, data security, user authentication and authorization are required. For smooth roaming, several studies have been made on a fast handover management in IPv6 Networks [18] and an integrated management that combines the strengths of

Mobile IP Location Registers (MIP-LR) and Session Initiation Protocol (SIP) [19]. As solutions for integrating 3G and WLAN services some of the recent studies have focused on a gateway [4], interworking techniques and architectures [5], a roaming and authentication service framework [24].

For location security and privacy, there were frameworks with a cryptographic approach of an authorized-anonymous-ID-based scheme [20] and algorithms for location discloser-control [21] and based on frequently changing pseudonyms [22]. However, in the 3G/WLAN interworking, the problem of location-aware efficient resource management has not been considered adequately in respect of reducing secure handover signaling as well as satisfying the QoS guarantees. As reconfigurable and adaptable features are indispensible in 4G networks, there are challenging issues with regard to location-aware seamless secure roaming.

3 Motivations and Requirements

In the vision of 4G mobile networks MN should be able to connect the best wireless networks among ad-hoc, personal, wired and wireless LANs and 3G mobile networks [8, 23]. However, the integration of these different networks generates new research challenges because of the heterogeneities of access technologies, network architectures, protocols and various service demands of mobile users [9]. Even with future mobile devices are likely equipped with reasonably accurate positioning capability, location-aware computing also includes many issues related to location privacy, consistent QoS guarantee, seamless vertical handover, common authentication, and so on. The following requirements should be considered to fulfill the promising services of the 3G/WLAN interworking.

- *Privacy and Security*: 3G/WLAN interworking shall not be compromised. It is required that authentication and key distribution should be based on the UMTS authentication and key agreement (AKA) procedure and EAP-AKA or EAP-SIM for WLAN [15,16]. The interworking system should eliminate the invasion of privacy by unwanted disclosure and commercial use of location information [2,20,21,22].
- *Global Secure Roaming*: MNs should be seamlessly served from foreign domain without any pre-established user authentication or authorization.
- *QoS guarantees*: The effectiveness of location-aware services depends not only on the user population but also on QoS guarantee. - Reduction of signaling overheads and latency of service delivery - Maintain QoS guarantees in different mobile systems.
- *Handover Management*: Handover latency is especially disruptive to continuous location tracking, even if most of the reauthentication during the handover in different networks are not lost but delayed.

To overcome the heterogeneities and to meet the requirements, a new common architecture with enhanced security, privacy and mobility management is indispensable to interconnect multiple access networks. The focus of this paper is trying to enforcing the security for seamless interworking of the new 3G/WLAN system by means of the efficient location-aware schemes.

4 Location-Aware Services with LBS Broker

In this section, we discuss how security can be improved through a new form of authentication based on location-aware mechanisms with LBS Broker. The location of the mobile device can be added as an additional authentication parameter for 3G/WLAN integration. Carefully managed location information could be used as authentication information. In order to provide location-aware seamless services at any time and anywhere, mobility is the key element in 3G/WLAN interworking systems. We have identified parameters for location-aware handover in consideration for vertical handover [10] as in Figure 1.

One of the difficulties in providing seamless roaming service is how to promptly and securely exchange security context during handover. The exchange of authentication information in advance by tracking and predicting users direction, so called *pre-warming* may helpful to simplify the authentication process during the handover to support seamless roaming service [11, 12]. For example, if the MN A in Figure 2 is on the subway, it will obviously handover from AP1 to AP2. Therefore, the authentication information of the mobile station needs to be delivered from AP1 to AP2 correctly in advance for seamless roaming before the handover procedure.

Reasonably accurate prediction of the user movements during the handover can help to solve the problem of the seamless secure interworking. A prediction of pre-warming zones is especially possible in track bounded wireless networks. In the case of the MN B, its pre-warming zones will include the areas around the subway station with areas along the track. In the case of the pedestrian Bob in Figure 2, its path may be effectively predicted by the history based location algorithms that has a record of the previous user movements and take into account the respective probability of movements together with factors such as the direction and the speed [12].

For seamless roaming service, we propose a LBS Broker including a location-aware authenticator that maintains a dynamic trust. Once LBS Broker predicts

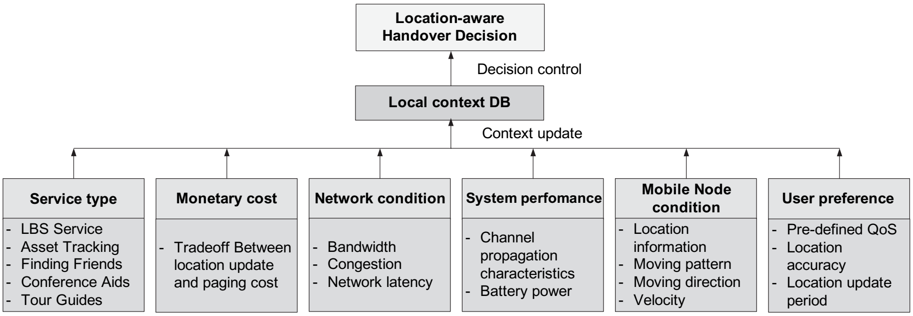
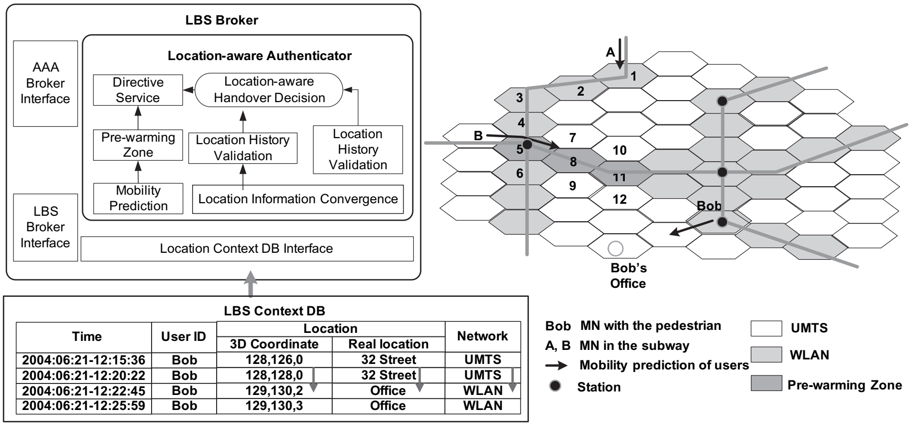


Fig. 1. The Proposed Location-aware handover decision criteria


LBS Broker
AAA Broker Interface
Location-aware Authenticator
Directive Service
Location-aware Handover Decision
Pre-warming Zone
Location History Validation
Location History Validation
Mobility Prediction
Location Information Convergence
LBS Broker Interface
Location Context DB Interface
LBS Context DB
A
B
1
2
3
4
5
6
7
8
9
10
11
12
Bob
Bob's Office
Bob MN with the pedestrian
A, B MN in the subway
Mobility prediction of users
Station
UMTS
WLAN
Pre-warming Zone

Time	User ID	Location		Network
		3D Coordinate	Real location	
2004:06:21-12:15:36	Bob	128,126,0	32 Street	UMTS
2004:06:21-12:20:22	Bob	128,128,0	32 Street	UMTS
2004:06:21-12:22:45	Bob	129,130,2	Office	WLAN
2004:06:21-12:25:59	Bob	129,130,3	Office	WLAN

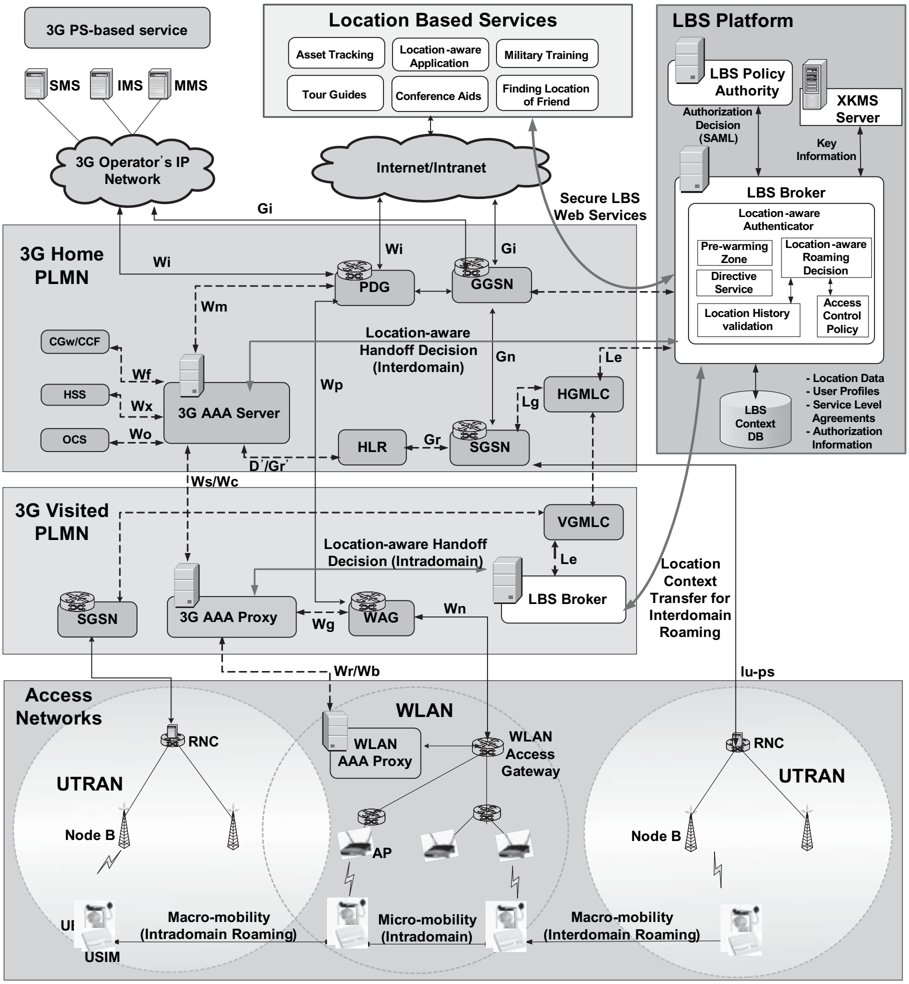
Fig. 2. LBS Broker with Location-aware Authentication and Pre-warming

a user's direction in heterogeneous networks, it forms a pre-warming zone to pre-establish trust relationships with AAA for fast handover. During the handover location history of the user in 3G networks could enforce the location-aware authentication. In the case of the roaming from UMTS to WLAN, LBS Broker validates the location history of the user then sends authentication and key information like Pairwise Master Key (PMK) to APs in the pre-warming zone in WLANs. Since there is not necessary to process full reauthentication, handover latency can be greatly decreased by performing only association and 4-way handshake (if necessary). If the MN B in Figure 2 is on the subway, LBS Broker decides APs within its pre-warming zone (5→ 8→11) and performs pre-authentication in advance for fast handover.

5 The 3G/WLAN Interworking System for Seamless Roaming Services with Location-Aware Authentication

5.1 3G/WLAN Interworking System with Location Services

We designed the location-aware security architecture for 3G/WLAN interworking to meet the location-aware computing requirements in section 3. Figure 3 shows the proposed interworking architecture. For satisfying the key requirements of 3G/WLAN interworking scenario 3 in Section 1, the user data traffic in WLANs is routed to the 3G home Public Land Mobile Network or visited PLMN through a component called a packet data gateway (PDG) or a wireless access gateway (WAG), which in the case of roaming is located in the preferred 3G visited PLMN. For several interfaces, Wn, Wm, Wi, Wg, and Wp, we also adopt the notation and functionality specified in [13]. Location services (LCS) is logically implemented on the network structure through the addition of one network node, the Gateway Mobile Location Center (GMLC) or Mobile



Wr/Wb: This interface carries AAA signaling between the WLAN and the 3G visited or home PLMN in a secure manner
Ws/Wc: This interface provides the same functionality as Wr/Wb but runs between a 3G AAA proxy and a 3G AAA server
Wn: This interface is used for transporting tunneled user data between the WLAN and the WAG
Wx: This reference point provides communication between AAA infrastructure and HSS
Wg: An AAA interface between the 3GPP AAA proxy and the WAG for provisioning of routing enforcement functions for authorized users
Wo: This is used by a 3GPP AAA server to communicate with the online charging system (OCS) for charging information
Wf: The interface between 3GPP AAA server and charging gateway function (CGF)/charging collection function (CCF) for charging
Wi: Reference point between the packet data gateway and a packet data network (external public or private)
D'/Gr': This optional interface is used for exchanging subscription information between the 3G AAA server and the HLR

Fig. 3. The 3G/WLAN Interworking System for Seamless Roaming Services with Location-aware Authentication(dashed lines: signaling; solid lines: data and signaling)

Location Center (MLC)[14]. LBS Broker may get location information directly from Gateway Mobile Location Center (GMLC).

5.2 Location-Aware Authentication and Secure Roaming Services

We present LBS Broker to securely provide the functionalities of location authority that gives a space of possible locations and can respond to queries on distances, routes, and proximity. LBS Broker plays key roles not only in location-aware authentication for fast roaming and but also in protecting users privacy.The agent model of LBS Broker could separate the security system from the location-based application. Supporting abstraction has a number of advantages. Firstly, it allows security unaware applications to be secured. This means that applications do not need to know much about the security features, because location security and privacy policies are enforced by LBS Broker. As an abstraction for location-aware computing, LBS Broker supports Web Services for interoperability of LBS. It also provides Web Services Security Specification like XML Signature, XML Encryption and Security Assertion Markup Language (SAML). LBS Broker act as a Policy Enforcement Point (PEP) that checks permission with the LBS policy authority, the Policy Decision Point (PDP) by requesting SAML assertion before making decisions and releasing the secured location data to the LBS service providers.

Intradomain (micro-mobility) authentication, authorization, and accounting (AAA) are handled together with LBS Broker and AAA server. When a user moves into a foreign network for interdomain roaming (macro-mobility), LBS Broker, AAA proxy and AAA server take charge of reauthentication process. LBS Broker enforce fast vertical handover by validation location history of MN using concepts of direction of user and pre-warming zone for 3G/WLAN interworking. Between AAA proxy in visited PLMN and AAA server in home PLMN, AAA context transfer protocol is used. Location context are exchanged between LBS Brokers. In the absence of context transfer, there may be large delays because of the network signaling required to re-establish QoS flows, re-authenticate the mobile user [10].

5.3 AAA Signaling for Location-Aware Authentication and Roaming

One of the challenging problems in the 3G/WLAN interworking systems is the interdomain roaming, which requires additional efforts to establish security relationships from the previous to the new access router in the different types of networks. To enforce the authenticity of MNs the location-aware authentication and secure roaming should be associated with AAA procedures.

The proposed location-aware secure roaming procedure from UMTS to WLAN is depicted in Figure 4. Based on the network advertisement data, a MN selects a preferred 3G visited PLMN and forms a second Network Address Identifier (NAI) corresponding to this PLMN [5]. The WLAN routes the AAA message to the 3G AAA Server or 3G AAA Proxy based on the NAI and the access authentication is performed. If the 3G AAA Server uses Location-aware authentication, it sends a location authentication request to LBS Broker. LBS Broker authenticates to the user and it sends a Location authentication response to the 3G AAA server. If the 3G AAA authenticates to the MN successfully, it sends EAP-Success message with keying material and some filter rules.

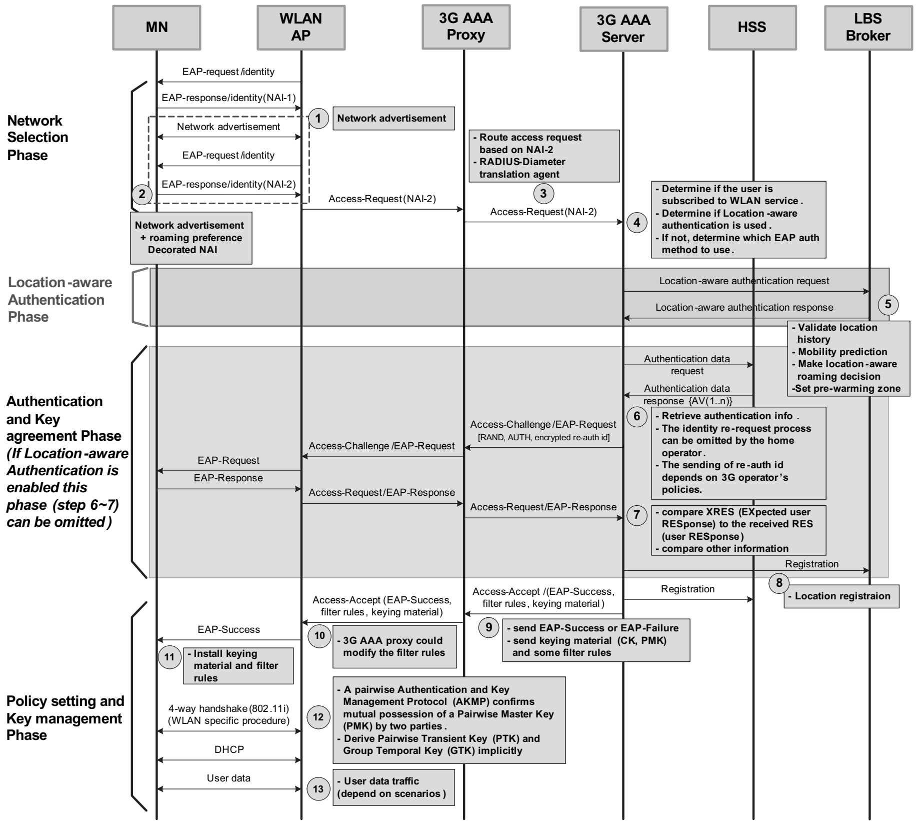

MN
WLAN AP
3G AAA Proxy
3G AAA Server
HSS
LBS Broker
EAP-request/identity
EAP-response/identity(NAI-1)
1
Network advertisement
Network Selection Phase
Network advertisement
EAP-request/identity
EAP-response/identity(NAI-2)
2
- Route access request based on NAI-2
- RADIUS-Diameter translation agent
3
Access-Request(NAI-2)
Access-Request(NAI-2)
4
- Determine if the user is subscribed to WLAN service.
- Determine if Location-aware authentication is used.
- If not, determine which EAP auth method to use.
Network advertisement + roaming preference Decorated NAI
Location-aware Authentication Phase
Location-aware authentication request
Location-aware authentication response
5
- Validate location history
- Mobility prediction
- Make location-aware roaming decision
-Set pre-warming zone
Authentication and Key agreement Phase
(If Location-aware Authentication is enabled this phase (step 6~7) can be omitted)
Authentication data request
Authentication data response {AV(1..n)}
6
- Retrieve authentication info.
- The identity re-request process can be omitted by the home operator.
- The sending of re-auth id depends on 3G operator's policies.
Access-Challenge/EAP-Request [RAND, AUTH, encrypted re-auth id]
Access-Challenge/EAP-Request
EAP-Request
EAP-Response
Access-Request/EAP-Response
Access-Request/EAP-Response
7
- compare XRES (EXpected user RESponse) to the received RES (user RESponse)
- compare other information
Registration
8
Registration
- Location registraion
Access-Accept /(EAP-Success, filter rules, keying material)
Access-Accept (EAP-Success, filter rules, keying material)
9
- send EAP-Success or EAP-Failure
- send keying material (CK, PMK) and some filter rules
EAP-Success
10
- 3G AAA proxy could modify the filter rules
11
- Install keying material and filter rules
Policy setting and Key management Phase
4-way handshake(802.11i) (WLAN specific procedure)
12
- A pairwise Authentication and Key Management Protocol (AKMP) confirms mutual possession of a Pairwise Master Key (PMK) by two parties.
- Derive Pairwise Transient Key (PTK) and Group Temporal Key (GTK) implicitly
DHCP
User data
13
- User data traffic (depend on scenarios)

Fig. 4. Location-aware Secure Roaming procedure from UMTS to WLAN

6 Experimental Results

In order to test our location-aware secure roaming in 4G networks, we created the testbed shown in Figure 5. We consider UMTS AKA and EAP-AKA for 3G/WLAN interworking. We also consider WLANs authenticate a MN according to the IEEE 802.1x and IEEE 802.11i standards which use EAP-TLS and EAP-TTLS. Table 2 summarizes the base parameters underlying the performance experiments. LBS Broker and LBS Policy Authority are running on server machines of Pentium III 933 MHz CPUs with Solaris 8 operating system (O/S). AAA Server is running on a server of Pentium III 800 MHz with Linux O/S and the modified FreeRADIUS library for RADIUS functionality. The cryptographic library is OpenSSL 0.9.7a and SAML Library is OpenSAML 0.9.1.

Figure 6 and 7 show the delay performances for the proposed location-aware secure roaming for micro-mobility between WLANs and for macro-mobility among CDMA, GPRS, UMTS and WLAN, respectively. The solid curves represent the measurements without our location-aware scheme and the dotted curves represent our location-aware case. We notice an important difference between the ex-

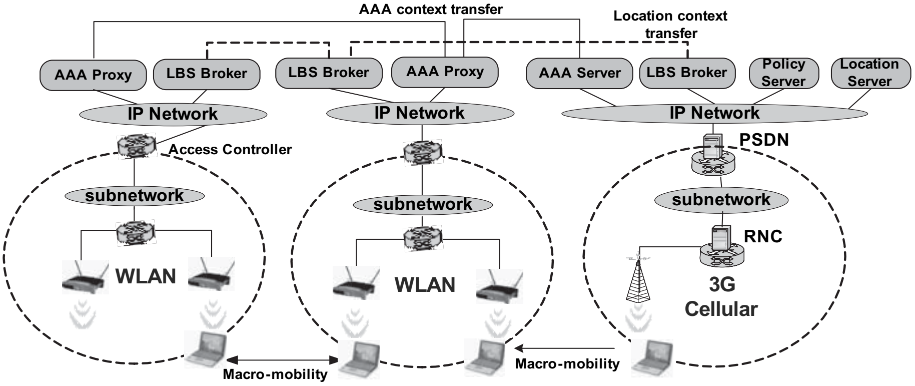


Fig. 5. Testbed of the Location-aware Security Architecture for UMTS/WLAN interworking

Table 2. Simulation Parameters of the Testbed of Location-aware Secure Roaming

Entity	Operation	Description	Performance
MN-AAA	802.1X full authentication (EAP-TLS)	Average delay	1,600ms
LBS Broker	Location history request to Location Server	Request location history of MN	50ms
LBS Broker	Location-aware authentication	Validation of location history of MN	80ms
LBS Broker	SAML Authorization Request	XML Parsing & RSA 1024 signature	27.4 ms
LBS Policy Authority	SAML Authorization Response	XML Parsing & RSA SHA-1 1024 bit key signature verify	20.4 ms
LBS Policy Authority	SAML Authentication Token generation and response to MN	3DES Symmetric key encryption	7.702 MB/s
LBS Broker	Response with Location information	RSA encrypt on 512 bit keys	31.201 KB/s
MN-AP	802.11 scan (active)	Average latency	40 300ms
MN-AP	802.11 reassociation with IAPP	Average latency	40ms
MN-AP	Fast Handover(4-way handshake only)	Average delay	60ms
802.11/CDMA	TCP parameter adjustment	Average delay	5000ms
802.11/GPRS	TCP parameter adjustment	Average delay	20000ms
UMTS/802.11	Intradomain UMTS to WLAN Handover with EAP-SIM authentication	Average delay	9,300ms

isting authentication case and the location-aware secure roaming case. When the moving MNs are increasing, our location-aware scheme does not create much burden on roaming as to selection of 802.1X authentication methods. In the roaming case of Figure 6, the location-aware scheme with EAP-TTLS CHAP shows almost the same performance with roaming of EAP-TLS without location-aware scheme. The overhead of our location-aware scheme in UMTS-to-WLAN roaming is 9.54% in Figure 7. We should consider trade-off among QoS of location-aware services including location update period and location precision, security and privacy. In fact, both dimensions of security strength and network performance are equally important, and achieving a good trade-off between two extremes is one fundamental challenge in security design for location-aware computing.

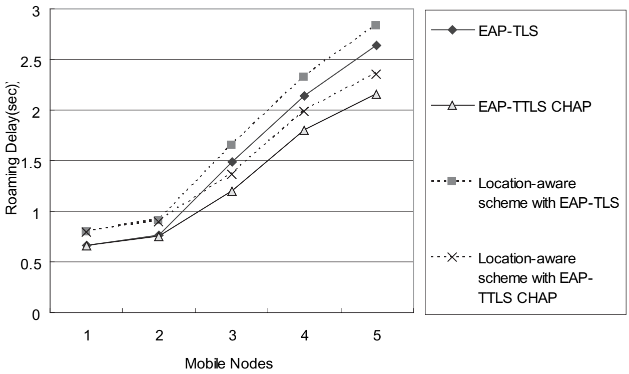


Fig. 6. Delay performance of Location-aware Secure Roaming from WLAN to WLAN

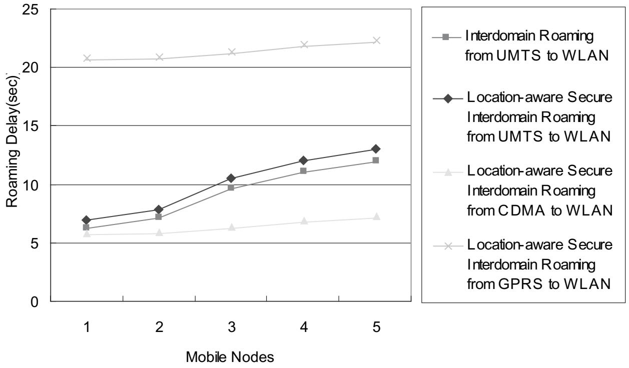


Fig. 7. Delay performance of Location-aware Secure Roaming from 3G to WLAN

7 Conclusions

We analyze mobility, interworking and security issues in location-aware computing and give our view on the future prospects for 3G/WLAN interworking. We introduce concepts of location-aware security and propose solution with several existing and new systems. The 3G/WLAN interworking system can be enhanced by new functionalities such as more advanced LBS service support by enabling efficient support of location-aware secure fast roaming with LBS Broker and location privacy policy control functions with LBS Policy Authority. This integrated scheme could provide the desired security features and requirements for survivable heterogeneous wireless networks. A testbed has been developed and experimental results of the secure handover have been presented. The proposed location-aware handover mechanism is currently being integrated with our secure Web Services infrastructure and 3GPP/WLAN interworking systems [2, 17].

References

1. Mike Hazas, et. al: Location-Aware Computing Comes of Age. IEEE Computer, Feb. 2004.
2. Minsoo Lee, Jintaek Kim, Sehyun Park, Jaeil Lee and Seoklae Lee: A Secure Web Services for Location Based Services in Wireless Networks. Lecture Notes in Computer Science, vol. 3042. May, 2004, pp. 332-344.
3. 3GPP TR 22.934: Feasibility Study on 3GPP System to WLAN Interworking,R6.
4. Feng, V. W.-S., et. al: WGSN: WLAN-based GPRS Environment Support Node with Push Mechanism. The Computer Journal, vol. 47, 2004. pp. 405-417.
5. A. K. Salkintzis:Interworking Techniques and Architectures for WLAN/3G Integration toward 4G Mobile Data Networks.IEEE Wireless Communications,June 2004.
6. D. E. Denning and P. D. MacDoran: Location-Based Authentication: Grounding Cyberspace for Better Security. Computer Fraud and Security, February 1996.
7. Jakob E. Bardram, et. al: Context-Aware User Authentication-Supporting Proximity-Based Login in Pervasive Computing. UbiComp'03, 2003.
8. Petri Mähönen, et. al: Hop-by-Hop Toward Future Mobile Broadband IP. IEEE Communications Magazine, March 2004.
9. Akyildiz, I.F., et. al.: A survey of mobility management in next-generation all-IP-based wireless systems. IEEE Wireless Communications, Aug. 2004.
10. McNair, J. Fang Zhu: Vertical handoffs in fourth-generation multinetwork environments. IEEE Wireless Communications, June 2004.
11. Christian Prehofer, et. al: Active Networks for 4G Mobile Communication: Motivation, Architecture and Application Scenarios. LNCS, vol. 2546. 2004.
12. R. Chellappa-Doss, A. Jennings and N. Shenoy: User Mobility Prediction in Hybrid and Ad Hoc Wireless Networks. ATNAC, Dec. 2003.
13. 3GPP TS 23.234 v6.0.0:3G System to WLAN Interworking;System Description,R6.
14. 3GPP TS 23.271:Functional stage 2 description of Location Services(LCS)",R6.
15. 3G TS 33.234 v050: 3G Security;Wireless Local Area Network(WLAN) Interworking Security.R6.
16. Koien, G.M.; Haslestad, T: Security aspects of 3G-WLAN interworking. IEEE Communications, November 2003.
17. Minsoo Lee, Jintaek Kim, Sehyun Park, Ohyoung Song and Sungik Jun: A Location-Aware Secure Interworking Architecture Between 3GPP and WLAN Systems, accepted for publication in Lecture Notes in Computer Science, Proc. ICISC 2004, Dec 2004.
18. Nicolas Montavont, et al:Handover Management for Mobile Nodes in IPv6 Networks.IEEE Communications, August 2002.
19. K. Daniel Wong, et. al: Mobility Management Scheme for Auto-configured Wireless IP Networks. IEEE Wireless Communications, October 2003.
20. Qi He, et. al: The Quest for Personal Control Over Mobile Location Privacy. IEEE Communications, May 2004.
21. MARCO GRUTESER et. al: Protecting Privacy in Continuous Location-Tracking Applications. IEEE SECURITY & PRIVACY, March-April 2004.
22. Alastair R. Beresford and Frank Stajano: Location Privacy in Pervasive Computing. IEEE PERVASIVE computing, Jan-Mar 2003.
23. Marques, V. et. al: An IP-based QoS architecture for 4G operator scenarios. IEEE Wireless Communications, June 2003.
24. Minghui Shi, et al: IEEE 802.11 roaming and authentication in wireless LAN/cellular mobile networks. IEEE Wireless Communications, Aug 2004.

A New per-Class Flow Fixed Proportional Differentiated Service for Multi-service Wireless LAN*

Meng Chang Chen[1], Li-Ping Tung[2], Yeali S. Sun[3], and Wei-Kuan Shih[2]

[1]Institute of Information Science, Academia Sinica, Taipei, Taiwan
mcc@iis.sinica.edu.tw
[2]Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
{lptung, wshih}@cs.nthu.edu.tw
[3]Dept. of Information Management, National Taiwan University, Taipei, Taiwan
sunny@im.ntu.edu.tw

Abstract. In this paper, we propose a new per-CLAss Flow fixed proportional differentiated service model (CLAF) and a companion medium access control scheme for multi-service wireless LANs (WLANs). The scheme is based on the IEEE 802.11 framework. Different from conventional relative differentiated service, in CLAF, a fixed bandwidth quality ratio is guaranteed on per-class per-flow basis regardless of the traffic load of each service class. Specifically, each service class is assigned a number of separate coordination periods, proportional to the policy-based bandwidth quality ratio for class isolation. Each class is associated with its own contention window size which is dynamically adjusted in accordance with the number of flows in the class in such a way to minimize collision probability between flows of the same class. Simulations results of the CLAF performance as well as a comparison with IEEE 802.11e EDCF including the support of QoS-sensitive VoIP applications are presented. The results show that the proposed scheme outperforms EDCF and achieves better resource utilization efficiency. It can provide users a more predictive, affirmative service guarantees than conventional relative differentiated service like IEEE 802.11e EDCF.

1 Introduction

Wireless LANs (WLANs) are gaining significantly in popularity and being deployed at a rapid rate. The beauty of WLANs is that they are scalable with low entry cost. As IEEE 802.11–based WLAN penetrates further, its Quality of Service (QoS) support for multimedia applications such as VoIP and video streaming is important and critical to the success of wireless communications, especially to produce profitable business. In the past, several QoS mechanisms based on the IEEE 802.11 wireless LAN framework have been proposed. Most of them base their methods on tuning three different parameters: *a*) duration of the Interframe Space (IFS)[1][2][3][4]; *b*) length of the contention window (CW)[1][2][5][6][7][8]); and *c*) length of the backoff timer [3][9] to provide service differentiation among different service classes. The

* This research was partly supported by NSC under grants NSC-92-2213-E-002-088 and NSC 92-2213-E-001-011.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 382 – 394, 2005.

IEEE 802.11e EDCF [1] is an example that combines the first two approaches. The scheme provides a simple "*relative*" differentiation of bandwidth sharing between classes. The problems with such distributed priority-based approach are two folds. First, the channel bandwidth sharing ratio between classes is a function of total channel load and the number of active wireless stations in the network. Second, there is no traffic regulation and performance guarantees to flows within the same class. Third, the lower classes can experience starvation effects if no restriction is placed on the load of higher classes.

Schemes [4][7][9] were also proposed to provide proportional channel bandwidth sharing between multiple service classes. Better than 802.11e EDCF, these schemes have the merit of preventing lower priority traffic from access starvation. But in all these schemes, the contention probability increases as more flows requesting the same class of service. Other related works include some relative differentiated service models proposed for wired networks, such as strict prioritization[10], price differentiation[11] and capacity differentiation[12][13][14]. As illustrated in [15], relative differentiated service can not provide consistent service differentiation, because resource allocated to each service class does not reflect actual class load variation.

In this paper, we propose a new service mode called *per-CLAss Flow fixed proportional differentiated service model* (CLAF) and its wireless medium access control scheme. Different from conventional relative differentiated service model [10][11][12][13][14], CLAF provides a fixed proportion on the bandwidth sharing between multi-class flows. Here, a *flow* is a unidirectional sequence of packets uniquely identified by the IP addresses and port numbers of the source and destination stations, as well as the IP protocol type. For example, suppose there are two service classes. Class 1 has one flow and Class 2 has two flows. In the case that the per-class flow bandwidth quality ratio policy is 2:1, according to the CLAF service model, the Class 1 flow will receive 1/2 of the channel capacity and each Class 2 flow will receive 1/4 of the channel capacity. The bandwidth share ratio of Class 1 flow and Class 2 flow is 2:1.

The proposed medium access control scheme for CLAF complies with the IEEE 802.11 framework. To achieve per-class flow fixed bandwidth share proportion in a multi-service wireless LANs, *separate* number of coordination periods are allocated to different service classes to achieve class isolation in channel access. Second, the number of coordination period allotted to a service class is proportional to the class's bandwidth share defined in the policy-specified bandwidth quality ratio. During the coordination period, both upstream and downstream flows of the same class contend for packet transmission. Third, each service class is associated with a different class contention window size whose value is adaptive to the number of flows in the class. Distributed flows of the same class will follow the baseline CSMA/CA protocol to resolve contention. Under the scheme, flows of different service classes access the wireless channel in a distributed, coordinated way. The advantages of CLAF include class isolation, prevention of lower priority traffic from access starvation, and the adaptation of channel resource allocation to different service classes based on their actual traffic load.

A common problem with the IEEE 802.11 CSMA/CA-based medium access control scheme is that the scheme does not provide access point (AP) the capability in channel access that reflects the traffic load at the AP. This is especially crucial for many APs serving as Internet gateways as well as for networks with asymmetric uplink and downlink traffic pattern. In these environments, AP usually carries many more downlink flows than any other wireless station. It often results inefficient resource utilization and low overall channel throughput. In our proposed per-class flow fixed proportional differentiated service, this problem is resolved by allocating channel resources on per-class, per-flow basis. Moreover, such allocation is in accordance with the actual number of flows in the network.

The paper is organized as follows. In Section 2, the proposed *per-CLAss Flow fixed proportional differentiated service model* (CLAF) is presented. In Section 3, the medium access control scheme is described in detail. In Section 4, the simulation results of the CLAF performance are presented. Then, we compare the results with IEEE 802.11e EDCF, including the support of QoS-sensitive VoIP applications. Section 5 gives the conclusion.

2 CLAF - The per-CLAss Flow Fixed Proportional Differentiated Service Model

In this section, we present the per-CLAss Flow fixed proportional differentiated service model for wireless local area networks. Different from conventional relative differentiated service model, CLAF provides fixed bandwidth share proportion to individual flow of different service classes. How bandwidth share between flows is defined in a system parameter called *bandwidth quality ratio* which is set by the network administrator as a resource management policy.

The CLAF service model is formulated as follows. Consider a wireless local area network supporting K service classes. Let φ_k be the target bandwidth quality ratio parameter; and ω_k be the target throughput measurement of a class k flow. The model imposes constrains of the following form for all classes:

$$\omega_1 : \omega_2 : ... : \omega_K = \varphi_1 : \varphi_2 : ... : \varphi_K \tag{1}$$

where $\varphi_1 > \varphi_2 > ... > \varphi_K$. The higher classes have the larger bandwidth shares. Let the total number of flows in service class k is N$_k$, k=1, …, K. Given the invariant bandwidth quality ratio, the aggregate throughput of each service class changes as the number of flows admitted to the class varies. We have the aggregate bandwidth quality ratio of the K service classes, denoted as $\{B_k\}$, as follows.

$$B_1 : B_2 : ... : B_K = N_1 \times \varphi_1 : N_2 \times \varphi_2 : \mathrm{K} : N_K \times \varphi_K \tag{2}$$

For example, consider a wireless network supporting two service classes with $\varphi_1 : \varphi_2 = 4:1$. The policy says that every Class 1 flow would be guaranteed four times as much the bandwidth share as of a Class 2 flow. If there are two Class 1 flows and three Class 2 flows in the network, the aggregate bandwidth share ratio would be 8:3.

3 The Medium Access Control Scheme

The medium access control scheme for the CLAF service model consists of three parts: *a*) the baseline channel access procedure for the contending flows of the same service class; *b*) the coordination of the channel access between different classes; and *c*) the protocol for the join and leave of a flow.

3.1 Baseline Intra-class Channel Access Procedure

In CLAF, the medium access control procedure for flows of the same class follows the basic backoff procedure as defined in the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol of the IEEE 802.11 Distributed Coordination Function (DCF) mechanism. When a station has a packet to transmit, it generates the random backoff timer from the range of 0 to the current *Contention Window-1*. The backoff timer is decremented while the wireless medium is sensed idle and is frozen when it is busy. When the backoff timer counts down to zero, the station transmits the packet. If two or more stations transmit at the same time, collision occurs. In our scheme, a collided station does *not* double contention window size for retransmission. The flow waits for the next coordination period of the class it belongs to retransmit the packet.

3.2 Inter-class Channel Access Scheme

3.2.1 The Size of the Class Contention Window

A new distributed coordination algorithm is used to regulate the channel access between classes. First, each service class is associated with a *class contention window*, denoted as $CW_k(t)$ whose size is determined based on the number of flows in the class at time *t* as follows:

$$CW_k(t) = CW_0^{\varepsilon}(N_k(t)) \tag{3}$$

CW_0^{ε} is the base contention window function, where ε is the target maximum collision probability in a coordination period and $N_k(t)$ is the number of flows in class *k* at time *t*. Flows belonging to the same service class use the same class contention window size in their backoff timer computation. An example of CW_0^{ε} specification is given in Table 1.

Table 1. An example of the base contention window function CW_0^{ε}

Number of flows	1	2	3	4	5
$CW_0^{0.25}$	1	4	8	11	15
Number of flows	6	7	8	9	10
$CW_0^{0.25}$	18	22	25	29	32

3.2.2 The Class Frames and Coordination Periods

EQ(3) only specifies the range of the backoff time for flows of the same class. To achieve per-class flow fixed bandwidth quality ratio, two structures - *Class Frame*s and *Coordination Periods* - are used. First, the channel access time axis is divided into a sequence of *superframe*. Each superframe consists of K class frames, one for each service class. A superframe always begins with the class frame of the highest priority class. For the k^{th} service class, its class frame will consist of φ_k coordination periods. In the case that a service class has zero flow, its class contention window size will be set to zero. It results in zero duration of the corresponding class frame. In other words, the channel access will immediately continue to the next lower service class without waste of channel capacity. Within a class coordination period, flows of the corresponding class follow the basic intra-class channel access backoff procedure for packet transmission. Fig. 1 depicts the inter-class channel access structure.

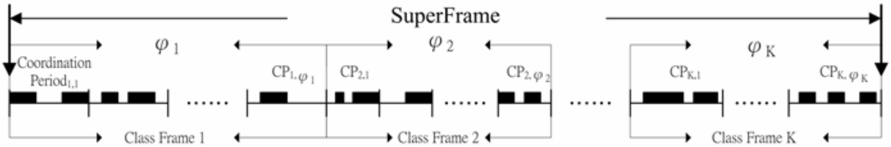


Fig. 1. The inter-class channel access structure assuming K service classes

3.3 Channel Access and Packet Scheduling at a Station

At the beginning of a superframe, the QoS AP broadcasts a Beacon message, including a list of the current class contention window sizes to use by wireless stations. The packet scheduling algorithm used at a wireless station is given in Fig. 2. In the algorithm, a backlogged flow makes one single channel access attempt in each coordination period of its associated class's Class Frame. Fig. 3 shows a possible implementation of the packet scheduling algorithm in the embedded kernel. The scheduler picks out the next packet from the flow with the minimum remaining backoff time to transmit in the currently-served service class and forwards the packet to the MAC layer. In the embedded kernel, software can be designed to effectively manage per-flow FIFO queueing and the associated backoff timers.

Fig. 4 is an example that illustrates the proposed scheme in intra- and inter-class channel access. There are two service classes with $\varphi_1 : \varphi_2 = 3:1$. There are two stations - A and B. Station A has two Class 1 flows ($f_{1,1}^A$ and $f_{1,2}^A$), and one Class 2 flow ($f_{2,1}^A$); and Station B has a flow of each service class ($f_{1,1}^B$ and $f_{2,1}^B$). Using the base contention window specification in Table 1, we have the Class 1 and 2 contention window size as CW$_1$=8 and CW$_2$=4, respectively. The backoff times of these flows at both stations are given in Table 2. Note that in this case the Class 1 contention window size is eight. Therefore, each Class 1 coordination period spans eight idle time slot duration to coordinate the start and end of a coordination period of a service class. This is necessary because in wireless LAN, each station does not know the actual number of backlogged flows of each service class in the other stations but individual class contention window sizes. Even though they know, if there is a collision, it is difficult to infer how many stations were involved in the collision. Therefore, the contention window size in terms of number of idle time slots is used for distributed coordination period synchronization. All stations monitor the channel events and count the number of idle slots to execute the proposed CLAF channel access scheme in a distributed and coordinated fashion.

```
Station_ Packet _Scheduling_and_Channel_Access_ Algorithm
/* used by a wireless station to determine packet scheduling sequence of flows of multiple classes in
each superframe */

On receiving the Beacon Message {
  for (i =1 to K)   {   /* For each Class Frame $i$ */
     for (j = 1 to  $\varphi_i$)   {   /* for each Coordination Period of Class $i$ */
           $b_{prev} = 0$ ; /* previous backoff timer */
           $\Gamma = \phi$ ; /* initialize un-served backlogged flow set */
           if ( $B_i \neq \phi$ )      /* $\mathrm{B_i}$ : the set of locally backlogged flows of Class $i$. */
                  Generate a different random backoff time for each flow in $\mathrm{B_i}$,
                  i.e.  $\Gamma = \{b_{i,m}\}$   $\forall m \in B_i$   /* $b_{i,m}$: backoff timer of flow $m$ of class $i$ */
           stationBackOffTime = 0; /* initialize station's backoff timer */
           while ( $\Gamma \neq \phi$ ) {
                  Select a flow $x$ such that  $b_{i,x} = \min\{b_{i,m} \mid b_{i,m} \in \Gamma\}$ ;
                  stationBackOffTime = ( $b_{i,x} - b_{prev}$ );
                  Follow the baseline channel access procedure;
                  When stationBackOffTime counts down to zero {
                        Transmit the head-of-line packet from flow $x$ to the network;
                        /* remove flow from un-served backlogged flow set */
                        $\Gamma = \Gamma \setminus \{b_{i,x}\}$ ;
                        $b_{prev} = b_{i,x}$ ;
                  } /* end of when */
           } /* end of while */
           /* All backlogged class $i$ flow have been served. */
           stationBackOffTime = $\mathrm{CW_i} -$  $b_{prev}$ ;/* remaining coordination period */
           Follow the baseline channel access procedure;
           Count down stationBackOffTime to zero;
     } /* end of j
  } /* end of i
}
```

Fig. 2. The packet scheduling algorithm for channel access at a wireless station

In a coordination period, if a collision occurs (as occurred in $\mathrm{CP}_{1,2}$ in Fig. 4), all collided flows will retransmit their packets in the next coordination period. If the collision takes place in the last coordination period of a Class Frame, retransmission will be deferred to the next superframe. Hence, in CLAF, there is *no* exponential backoff time computation for packet retransmission.

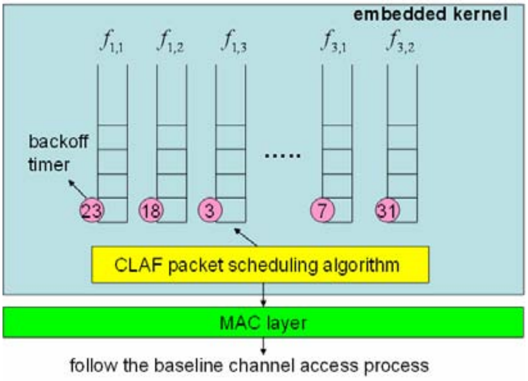


Fig. 3. The implementation architecture of the packet scheduling and queueing at a wireless station in the CLAP service model

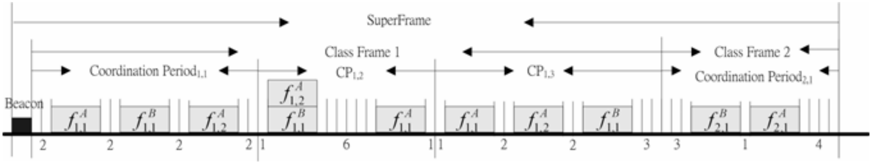


Fig. 4. The packet scheduling sequence and channel access events

Table 2. Backoff times of the example in Fig. 4

Station	A			B	
Class	Class 1		Class 2	Class 1	Class 2
Flow Index	$f_{1,1}^A$	$f_{1,2}^A$	$f_{2,1}^A$	$f_{1,1}^B$	$f_{2,1}^B$
Backoff Time	2	6	4	4	3
	7	1	-	1	-
	1	3	-	5	-

3.4 Procedure for Flow Join and Leave

In CLAF, the class contention window size is dynamically adjusted based on the number of flows of the service classes. To keep track of the number of flows, a control frame is used. As shown in Fig. 5, each superframe is followed by a control frame in which stations send Re-association Requests to the QoS AP specifying the number of new flows to join and the number of flows to leave for each service class.

The QoS AP will reply a Re-association Response to the station with a status code indicating whether the join request is accepted or not. Based on the results, the QoS AP computes new class contention window sizes, if necessary and updates all wireless stations in the next Beacon message.

No stations are allowed to send Association and Re-association messages in a superframe. During the control frame, stations use a separate contention window size denoted as $ControlFrameContenetionWindow_{min}$ to randomize channel access backoff times when submitting the join/leave requests.

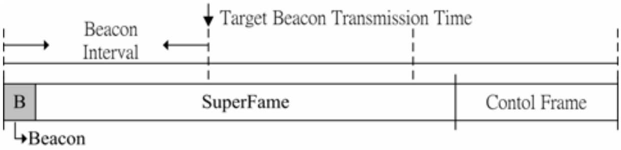


Fig. 5. A channel access round consists of a superframe for multi-class packet transfer plus a control frame for notifying flow join and leave

3.5 Base Contention Window Function

Given the number of flows of a service class, we want to find the contention window size ($CW_0^{\varepsilon}(n)$) such that the probability of colliding flows is upper bounded by a small value ε in a coordination period. The computation is based on the *Inclusion-Exclusion Principle* and is summarized as follows:

Step 1: Compute the mean collision number by using $n \times \varepsilon$

Step 2: Compute the expected number of collided flows $E_n^{CW}[Coll] = \sum_{i=0, i\neq 1}^{n} p_n^{CW}(i) * i$ under different pair (CW, n), where $CW \geq n$. Here,

$$p_n^{CW}(i) = \begin{cases} \dfrac{F(CW,n)}{CW^n}, & i = n, \quad i \neq 1 \\ \dfrac{C_k^n C_k^{CW} k!(F(CW-k, n-k))}{CW^n}, & 2 \leq i \leq n-1, \quad k = n-i, \quad n \geq 3 \end{cases} \tag{4}$$

and

$$F(CW,n) = CW^n - \sum_{i=1}^{n} (-1)^{i-1} C_i^n C_i^{CW} i!(CW-i)^{n-i} \tag{5}$$

Step 3: To get $CW_0^{\varepsilon}(n)$, we can find smallest *w* that satisfies $E_n^{CW}[Coll] < n \times \varepsilon$ from a sequences of $\{E_n^{CW}[Coll]\}$. Then $CW_0^{\varepsilon}(n) = w$.

4 Performance Evaluations

In this section, we provide a performance evaluation of the proposed per-CLAss Flow fixed proportional differentiated service model (CLAF) and its medium access control scheme via simulations. The simulator is implemented in C[16]. The parameter values of the baseline Intra-class Channel Access procedure used in the simulations are the same as those in 802.11 (see Table 3) in ns-2[17] configuration. Consider N wireless stations and a QoS AP. All communication between stations must be forwarded by the QoS AP. Packets are fixed of length 1K bytes.

Table 3. IEEE 802.11 PHY/MAC parameters used in sumulation

SIFS	10 µs	Slot_time	20 µs
DIFS	50 µs	ACK Size	14 bytes
MAC Header	28 bytes	PreambleLength	144 bits
PLCP Header Length	48 bits	Rate	11 Mbps

4.1 Fixed Proportional Bandwidth Differentiation

Fig. 7 shows the fixed bandwidth sharing proportion is enforced between per-class flows regardless of the traffic load of individual classes. Fig. 6 is the simulation configuration. There are three service classes with $\varphi_1 : \varphi_2 : \varphi_3 = 3:2:1$. Initially, the bandwidth quality ratio between Class 1 flow and Class 3 flow is approximately 3:1. The throughputs of flows of the same class are closely equal. At time 50, two Class 2 flows join the network. The resulting bandwidth sharing ratio changes to 3:2:1, while the actual throughput received by per-class flow is reduced due to the new flows. When the new flows leave, the throughput share restores to the initial state. The aggregate throughput of each service class is shown in Fig. 8, as the product of the class's bandwidth share proportion and the number of admitted flows in the class.

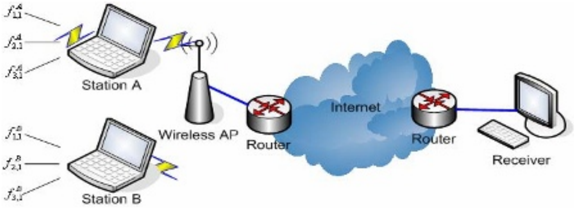


Fig. 6. The network configuration of Fig.7

4.2 Performance Comparison with IEEE 802.11e EDCF

To illustrate the performance difference between CLAF and EDCF, previous experimental scenario is repeated for EDCF using ns-2. The minimum contention window sizes for the Class 1, 2 and 3 are 16, 32 and 48, respectively. Fig. 9 shows the throughputs of flows of different classes. The throughput proportion is non-deterministic and has no direct relationship with the minimum contention window size. Moreover, throughput performance of individual flow fluctuates heavily.

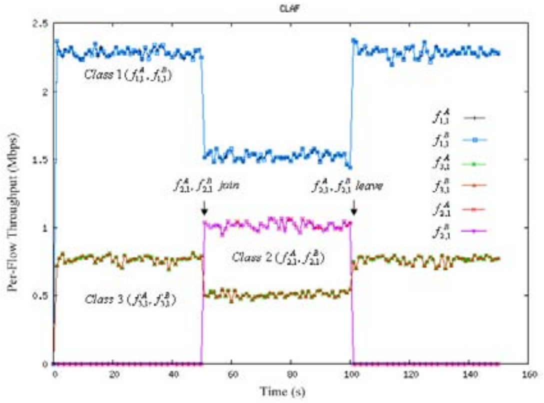


Fig. 7. The per-class flow throughput performance using CLAF ($\varphi_1 : \varphi_2 : \varphi_3 = 3:2:1$)

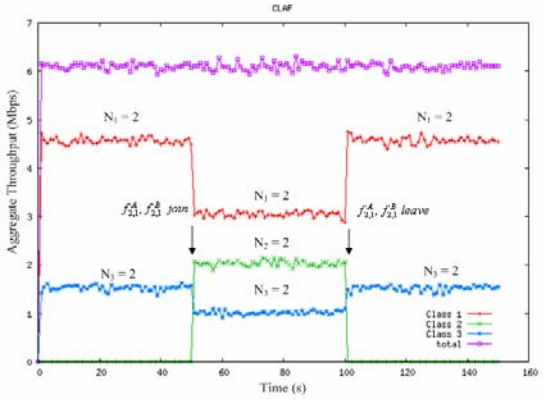


Fig. 8. The per-class aggregate throughput performance of Fig.7

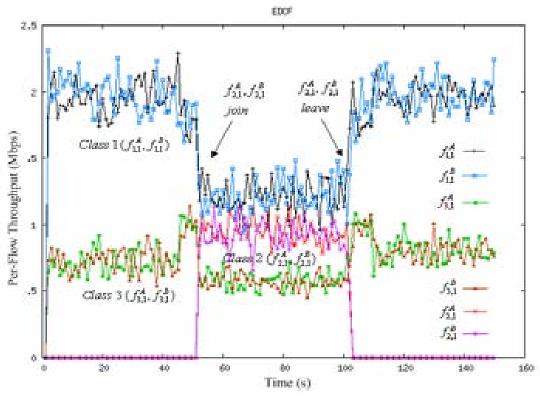


Fig. 9. The per-class flow throughput performance using EDCF ($\varphi_1 : \varphi_2 : \varphi_3 = 3:2:1$)

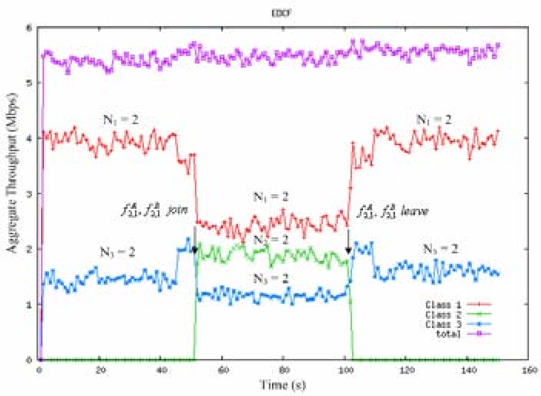


Fig. 10. The per-class aggregate throughput performance of Fig.9.

Comparing the result with Fig. 7, CLAF can guarantee a fixed bandwidth proportion to individual flow of different service classes and achieve more stable throughput performance at both the flow and class levels. Furthermore, the aggregate throughput of the EDCF is lower than that of the CLAF as shown in Fig. 10. It shows that CLAF is more efficient in the resource utilization.

4.3 VoIP over Wireless LAN

In this experiment, we want to show the merit of CLAF in the support of real-time multimedia applications. There are two kinds of VoIP calls - G.711 (Class 1) and G.729 (Class 2). Each call is between a wireless station and a wired host connected to a LAN on the other interface of the QoS AP. Hence, each VoIP call generates an

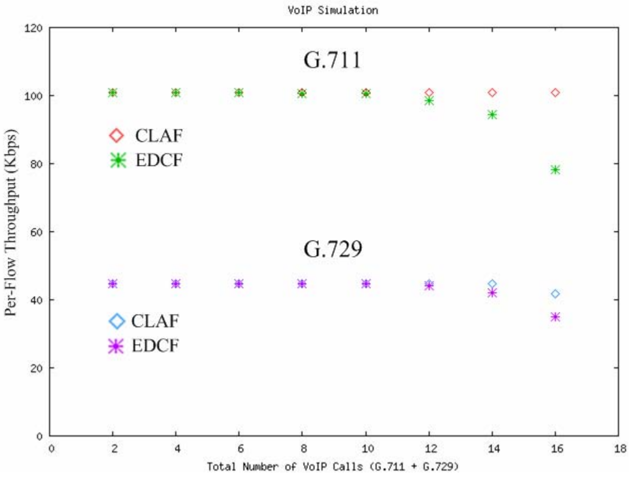


Fig. 11. The average throughput performance for VoIP using CLAF and EDCF

uplink and a downlink flow in the wireless LAN, one for each direction. The offered loads of the G.711 and G.729 VoIP flow are 100.8Kbps and 44.8Kbps, respectively (including RTP/UDP/IP/MAC packet header overhead). Hence, the bandwidth quality ratio is set to 9:4. In EDCF, both G.711 and G.729 flows belong to the same access category 3 (AC3), and the minimum and maximum contention window is 7 and 15 respectively.

Fig. 11 compares the average throughput of a VoIP flow between CLAF and EDCF under different numbers of VoIP calls. One can see that the throughput performance of both G.711 and G.729 flows are well protected under CLAF regardless the number of VoIP calls. For EDCF, the VoIP throughput performance is acceptable when there is no congestion within the class (i.e. less than ten calls). For CLAF, the maximum number of calls in this case is fourteen calls (i.e. 28 flows), more than ten calls for EDCF. CLAF can not only provide individual flow QoS but also achieves higher overall throughput performance.

5 Conclusions

In this paper, we have presented a new per-CLAss Flow fixed proportional differentiated service model (CLAF) and its medium access control scheme for multi-service wireless local area networks. Different from conventional differentiated services, CLAF provides a) policy-based fixed proportional differentiated service; b) such fixed proportional service differentiation is on per-class flow basis; and c) each class contention window size is adjusted to reflect the actual traffic load of the class. The simulation results show that the proposed scheme successfully provides per-flow fixed proportional differentiated service between multiple service classes. Second, we compare throughput performance between CLAF and IEEE 802.11e EDCF. The proposed scheme outperforms EDCF in terms of providing fixed proportional bandwidth share to individual flows of different service classes. It also achieves higher channel throughput. In the VoIP simulation runs, the scheme can as well better support real-time applications with QoS constraints. In summary, CLAF provides users a more predictive, affirmative service guarantees than conventional relative differentiated service like IEEE 802.11e EDCF.

Reference

1. IEEE 802.11e draft/D4.0, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS),” 2002
2. I. Ada and C. Castelluccia, “Differentiation Mechanisms for IEEE 802.11,” in *proceedings of INFOCOM* 2001
3. J. Deng and R. S. Chang, “A Priority Scheme for IEEE 802.11 DCF Access Method,” in *IEICE Trans. Communications*, vol. E82B, No. 1, 1999
4. W. Pattara-atikom, S. Banerjee and P. Krishnamurthy, “Starvation Prevention and Quality of Service in Wireless LANs,” in *Proceedings of WPMC* 2002

5. M. Barry, A.T. Campbell, and A. Veres, “Distributed Control Algorithm for Service Differentiation in Wireless Packet Networks,” in *Proceedings of INFOCOM* 2001.
6. K. Kim, A. Ahmad, and K. Kim, “A Wireless Multimedia LAN Architecture Using DCF with Shortened Contention Window for QoS Provisioning,” in *IEEE Communications Letters*, vol. 7, No. 2, February 2003
7. A. Banchs and X. pérez, “Providing Throughput Guarantees in IEEE 802.11 Wireless LAN,” in *Proceedings of WCNC* 2002
8. A. Banchs and X. pérez, “Distributed Weighted Fair Queuing in 802.11 Wireless LAN,” in *Proceedings of ICC* 2002
9. N. H. Vaidya, P. Bahl, and S. Gupta, “Distributed Fair Scheduling in a Wireless LAN,” in *Proceeding of ACM MOBICOM* 2000
10. C. Dovrolis and P. Ramanathan, “A Case for Relative Differentiated Services and the Proportional Differentiation Model,” in *IEEE Network*, Sep/Oct 1999
11. A. M. Odlyzko, “Paris Metro Pricing: The Minimalist Differentiated Services Solution,” in *Proceedings of IEEE/IFIP International Workshop on Quality of Service* 1999
12. A. Demers, S. Keshav, and S. Shenker, “Analysis and Simulation of a Fair Queueing Algoright,” in *Internetworking: Research and Experience,* pp. 3-26, 1990
13. A. K. Parekh, and R. G. Gallager, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case,” in *IEEE/ACM Trans. on Networking,* vol. 1, pp. 344-357, June 1993
14. J. C. R. Bennett and H. Zhang, “Hierarchical Packet Fair Queueing Algorithm,” in *IEEE/ACM Trans. on Networking,* vol. 5, pp. 675-689, Oct. 1997
15. C. Dovrolis, and D. Stiliadis, “Relative Differentiated Services in the Internet: Issues and Mechanism, “ in *ACM SIGMETRICS*, May 1999
16. Simulation source code, http://140.112.107.197/CLAF
17. “Network Simulator ns-2”, http://www.isi.edu/nsnam/ns/

Reservation and Grouping Stations for the IEEE 802.11 DCF

Yang Xiao[1], Haizhon Li[1], Kui Wu[2], Kin K. Leung[3], and Qiang Ni[4]

[1] Computer Science Division, University of Memphis,
373 Dunn Hall, Memphis, TN 38152, USA
yangxiao@ieee.org,hli1@memphis.edu
[2] Department of Computer Science, University of Victoria,
Victoria, British Columbia, Canada V8W 3P6
wkui@cs.uvic.ca
[3] Electrical and Electronic Engineering and Computing Departments,
Imperial College, London SW7 2BT, United Kingdom
kkleung@ieee.org
[4] Hamilton Institute, National University of Ireland, Maynooth
Qiang.Ni@ieee.org

Abstract. In this paper, we propose a novel contention-based protocol, called Reservation with Grouping Stations (RGS) for the IEEE 802.11 Distributed Coordination Function (DCF). In the proposed RGS scheme, stations are grouping into the active group, the ready group, and the idle group. Furthermore, stations announce the next frame's backoff counter beforehand. The proposed scheme is compared with the DCF and another scheme in the literature, and simulation results show that the proposed scheme is the best among these three schemes.

1 Introduction

The IEEE 802.11 Medium Access Control (MAC) employs a mandatory contention-based channel access function called Distributed Coordination Function (DCF), and an optional centrally controlled channel access function called Point Coordination Function (PCF) [1]. The DCF adopts a carrier sense multiple access with collision avoidance (CSMA/CA) with binary exponential backoff. In the DCF a station with a frame to transmit monitors the channel activities until an idle period, equal to a distributed inter-frame space (DIFS), is detected. After sensing an idle DIFS, the station waits for a random backoff interval before transmitting. The backoff time counter is decremented in terms of slot time as long as the channel is sensed idle. The counter is stopped when a transmission is detected on the channel, and reactivated when the channel is sensed idle again. The station transmits its frame when the backoff time reaches zero. At each transmission, the backoff time is uniformly chosen in the range [*0, CW-1*], where *CW* is the current backoff window size. At the very first transmission attempt, *CW* equals to the initial backoff window size CW_{min}. After each unsuccessful transmission, *CW* is doubled until a maximum backoff window size value CW_{max} is reached. Once it reaches CW_{max}, *CW* shall remain at the value CW_{max} until it

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 395 – 405, 2005.

is reset. *CW* shall be reset to CW_{min} after every successful attempt to transmit, or the retransmission counter reaches the retry limit L_{retry}. In the latter case, the frame will be dropped. After the destination station successfully receives the frame, it transmits an acknowledgment frame (ACK) following a short inter-frame space (SIFS) time. If the transmitting station does not receive the ACK within a specified ACK Timeout, or it detects the transmission of a different frame on the channel, it reschedules the frame transmission according to the previous backoff rules.

There have been many performance studies and performance enhancements for the DCF [2-14]. Calì et al. [2] studied an optimization method for a p-persistent Wireless LAN (WLAN) MAC, and proposed adaptive backoff algorithms for the p-persistent WLAN MAC [3]. Bianchi [4] proposed a simple and accurate analytical model to compute saturation throughput. Ziouva and Antonakopoulos [5] improved Bianchi's model to derive saturation delay. Bing [6] provided a performance analysis by a quantitative approach. Huang and Chen [7] gave approximate models that account for hidden terminals. Chhaya and Gupta [8] calculated the throughput of CSMA/CA using a simple model with the probabilities of capture and the presence of hidden stations. Tray and Chua [9] provided a good and approximate model for CSMA/CA. Xiao and Rosdahl [10, 11] identified a throughput upper limit for higher data rates and studied a group transmission and acknowledgement scheme. Leung et al. [12] studied the IEEE 802.11 MAC for outdoor cellular networks. Baldwin et al. [13, 14] proposed a real-time MAC protocol for ad hoc WLANs, in which two concepts are adopted, transmission deadline and a station's next backoff value (BV). The transmission deadline is used for selectively discarding frames which are late for their transmission deadlines at the MAC layer. A station's next BV is adopted to decrease collisions under a constant backoff window size, i.e., 8N and $(2 + \lfloor 6 / (R^{1/2}) \rfloor)N$ in [13] and [14], respectively, where N is the estimated number of stations in the network and R is the channel data rate in Mbps , and therefore, the scheme is called Enhanced Collision Avoidance (ECA).

As indicated in the most of studies [2-9], when the number of competing stations increases, the performance dramatically decreases. In this paper, we propose a novel scheme for contention-based MAC called Reservation with Grouping Stations (RGS) to reduce collisions and improve the system performance. The fundamental reason of causing collisions in the DCF is that stations do not know other stations' information such as backoff counters so that two stations with the same backoff counter do not know that they will be definitely collided sometime later. If such information can be known beforehand, collisions may be avoided by adopting different backoff counters, and unnecessary collisions and wasted waiting time may be avoided. Therefore, in the RGS scheme, backoff counters of next frames, if available, are generated beforehand and announced in frame transmissions. We further classify stations into three groups: the active group, the ready group, and the idle group. The motivation for grouping stations is that stations in the active group do not have collisions if there are no hidden nodes since their next frames' backoff counters are known by other stations which avoid use of conflicting backoff counters. Collisions may happen among stations in the ready group since their next frames' backoff counters have not been known by other stations yet. The hidden node problem will be handled by Request-To-Send (RTS)/Clear-To-Send (CTS) mechanism.

This paper is organized as follows. Section 2 describes the proposed RGS scheme in detail. Performance studies are conducted in Section 3 via simulations. We conclude this paper in Section 4.

2 Reservation with Grouping Stations

In this section, we propose a novel scheme for contention-based MAC called Reservation with Grouping Stations (RGS).

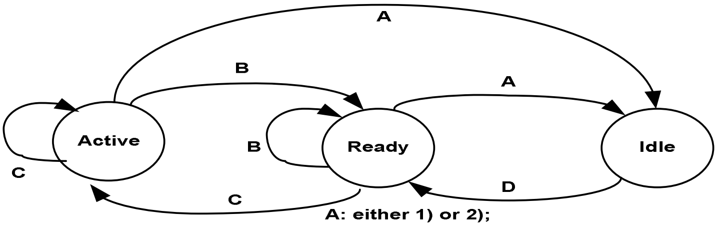
In the RGS scheme, backoff counters of next frames, if available, are generated beforehand and announced in frame transmissions. In other words, in a station's any frame transmission, referred to as the current frame transmission, if the next frame is ready, i.e., waiting in the queue to be transmitted, the next frame's backoff counter is generated beforehand and embedded in the current frame header to announce to other stations. If the current frame is successfully transmitted, other stations know the station's next frame's backoff counter. Otherwise, other stations have no way to know the station's next frame backoff counter.

Any station denoted as S has to maintain other stations' backoff counters locally in a table, called Backoff Counter Table (BCT), shown in Fig.1a. Fig.1a shows S's BCT, in which S's own backoff counter is 4, and backoff counters of stations A, B, and D are 6, 2, and 9, respectively. Station C had already transmitted its previous frame which does not include its next frame backoff information, and we use -1 to indicate the case that the next frame's backoff counter is not available. Whenever S hears an announcement of the next frame's backoff counter of another station denoted as F, S stores/updates F's backoff counter in S's local BCT. whenever S's backoff counter is decreased due to the channel access rule described in introduction such as detecting an idle slot, other stations' backoff counters in S's BCT are also decreased; whenever S's backoff counter freezes when detecting a busy channel, other stations' backoff counters in S's local table also freeze. Before S transmits the current frame, if it has the next frame waiting in the queue, it generates a random backoff counter for the next frame and embedded it in the current frame's MAC header to announce to other stations. If S is in the ready state defined in the next, whenever it receives a frame of the station F indicating that F has chosen the same backoff counter, S needs to choose another backoff counter since a collision will certainly occur.

Furthermore, as shown in Fig.1b, a station S has three states: active, ready, and idle. S is in the idle state if it has no frame ready to transmit. S is in the active state if it has a frame ready to transmit and this frame's backoff counter has been successfully announced through the previous successfully transmitted frame so that other stations know this information. S is in the ready state if it has a frame ready to transmit, but this frame's backoff counter has not been successfully announced to other stations. All the stations in the active state, the ready state, and the idle state form an active group, a ready group, and an idle group, respectively. We explain S's state diagram in Fig.1b as follows.

Stations	Self	A	B	C	D	■ ■ ■
Counters	4	6	2	-1	9	■ ■ ■

(a) Backoff Counter Table (BCT)



A: either 1) or 2);
B: either 3) or 4);
C: 5);
D: 6);

where

1) transmit successfully with an empty queue
2) collided/corrupted and retry limit is reached with an empty queue
3) collided/corrupted and retry limit is not reached;
4) collided/corrupted and retry limit is reached with non-empty queue
5) transmit successfully with a non-Empty queue
6) a frame arrives

(b) A station's state diagram

Fig. 1. The state diagram of a station S

a) S in the idle state changes to the ready state when a frame arrives (condition D);
b) S in the ready state changes to the active state if a frame, denoted as the current frame, is transmitted successfully and there is another frame, denoted as the next frame, in the queue so that the next frame's backoff is generated beforehand and announced through the current frame's transmission;
c) S in the ready state remains in the ready state if the current frame transmission is collided/corrupted and the retry limit is not reached;
d) S in the ready state remains in the ready state if the current frame transmission is collided/corrupted, the retry limit is reached, and there is another frame in the queue;
e) S in the ready state changes to the idle state if the current frame transmission is collided/corrupted, the retry limit is reached, and there is not another frame in the queue;
f) S in the ready state changes to the idle state if the current frame transmission is transmitted successfully and there is not another frame in the queue;
g) S in the active state remains in the active state if a frame is transmitted successfully and there is another frame in the queue;
h) S in the active state changes to the ready state if the current frame transmission is collided/corrupted and the retry limit is not reached;
i) S in the active state changes to the ready state if the current frame transmission is collided/corrupted, the retry limit is reached, and there is another frame in the queue;

j) S in the active state changes to the idle state if the current frame transmission is collided/corrupted, the retry limit is reached, and there is not another frame in the queue;
k) S in the active state changes to the idle state if the current frame transmission is transmitted successfully and there is not another frame in the queue;

Stations in the active group do not have collisions if there are no hidden nodes since their next frames' backoff counters are known by other stations which avoid using conflicted backoff counters. The hidden node problem can be reduced if Request-to-Send (RTS)/Clear-to-Send (CTS) mechanism is adopted. Collisions may happen among stations in the ready group since their next frames' backoff counters have not been known by other stations yet. If the hidden nodes exist, active stations may get collisions and in this case, the station will change from the active state into the ready state, as shown in Fig.1.

The next frame's backoff counter is chosen uniformly and randomly from the set $\{a_1, a_2 ..., a_n\}$, where $a_1 < a_2 <...< a_n$ such that for any integer m where $a_n \geq m \geq 0$, we have either $m \in \{a_1, a_2 ..., a_n\}$ or m equals to one of other stations' backoff counters in the station's BCT table. In other words, the set includes the next n available/unreserved slots. The purpose of the above choices is to avoid choosing conflicted slots already reserved by other stations.

Let N_1, N_2, and N_3 denote the numbers of stations in the active group, the ready group, and the idle group, respectively.

We propose an approach to estimate the value of N_2. Each station measures its activities during three states as follows. Let $T_1(j)$, $T_2(j)$, and $T_3(j)$ denote the portions of time spending in the active state, the ready state, and the idle state, respectively, for the station j. We have: $T_1(j) + T_2(j) + T_3(j) = 1.$ Whenever station j transmits a frame, it piggybacks the $T_1(j)$ and $T_2(j)$ values. Note that the $T_3(j)$ value can be derived based on the $T_1(j)$ and $T_2(j)$ values. Therefore, other stations also know them ($T_1(j)$, $T_2(j)$, and $T_3(j)$) and periodically update their values for station j. Let $M(\Delta t\ ,\ t)$ denote the number of piggybacked frames from different stations obtained during the time period from t - Δt to t, where Δt is a relative large value. Multiple frames from the same station count only once, i.e., the most current one. Then, current measured $N_2(t)$ value can be approximated by

$$N_{2,current}(t) = \sum_{j=1}^{M(\Delta t,t)} T_2(j) \tag{1}$$

Note that some idle stations may not necessarily send the piggyback frames, but this information does not have impacts on estimation of the value N_2. Let t' denote the previous measurement time, and α $(0 \leq \alpha \leq 1)$ denote a weight value. An exponential smoothing technique can be used to obtain estimated $N_2(t)$ value as follows.

$$N_2(t) = \alpha N_2(t') + (1-\alpha) N_{2,current}(t)\ . \tag{2}$$

In the proposed RGS scheme, the original binary exponential backoff is still used. However, the initial window size is chosen as $2(N_1 + N_2)$.

3 Simulation Results

In the simulations, we adopt the IEEE 802.11a as an example, and we compare the proposed scheme with the DCF and the ECA scheme in [13-14] in terms of average throughput, instant throughput, access delay, and the number of collisions.

Parameters are chosen as the default values in the IEEE 802.11a. The control rate is 24Mbps and the data rate 54Mbps. Beacon interval is 100ms. The mean of frame sizes is 300 bytes. Each station has traffic with ON-OFF periods. Initially there are 3 stations in the system. After each 5 seconds, 3 stations join the system until total 60 stations in the system. Each station operates in ON/OFF states alternatively. The times spent in the ON and OFF states are exponentially distributed with a mean of 3 seconds and 6 seconds, respectively. In the ON state, the station's queue is never empty. In the OFF state, the station has nothing to transmit. The total simulation time is 300s.

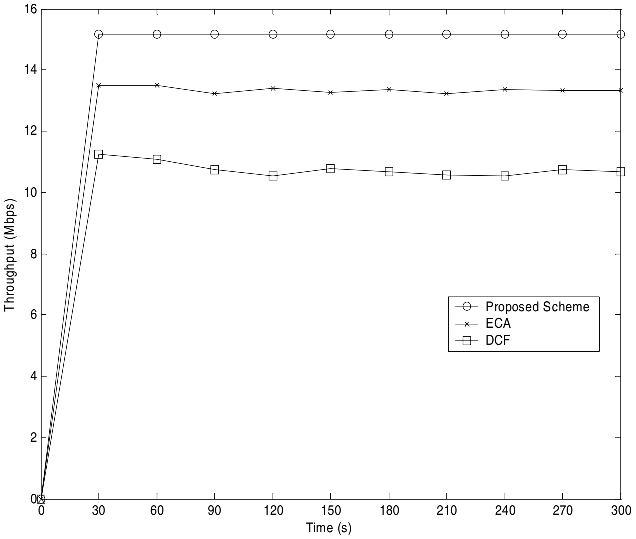


Fig. 2. Average Throughput (Mbps)

Fig. 2 compares the average throughputs of the DCF, ECA, and the proposed scheme. As illustrated in the figure, both the proposed scheme and the ECA outperform the DCF. The proposed scheme is the best among the three approaches in terms of the average throughput.

Fig. 3, Fig. 4, and Fig. 5 show the instant throughputs of the DCF, ECA, and the proposed scheme, respectively, where the instant throughput is defined as the throughput during a very short interval (per beacon interval), whereas the average throughput is defined as the throughput during a much larger interval. We observe that throughputs in the DCF and the ECA fluctuate a lot, whereas in the proposed scheme, throughput is much stable.

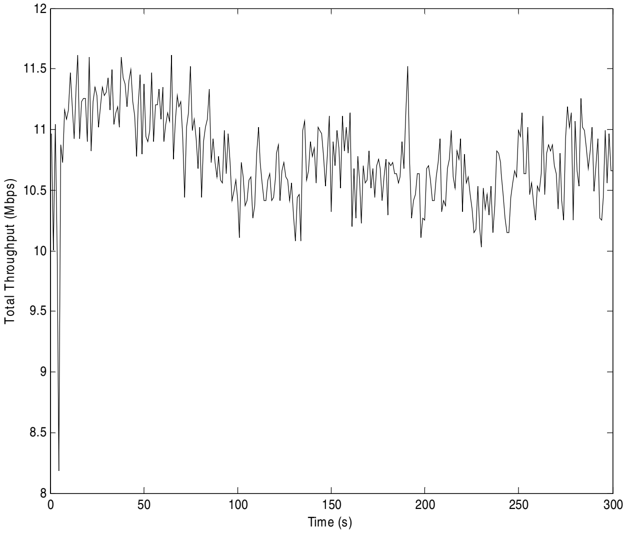


Fig. 3. Instant Throughput of the DCF (Mbps)

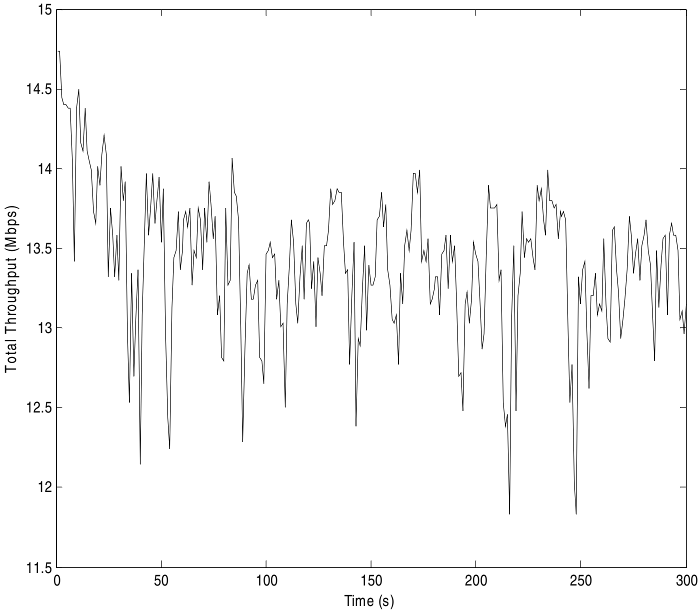


Fig. 4. Instant Throughput of the ECA (Mbps)

Fig. 6 and Fig. 7 show the numbers of collisions. As illustrated in Fig. 6, both the proposed scheme and the ECA have small numbers of collisions and are better than the DCF. The DCF in facts has a large number of collisions. Fig. 7 compares the proposed

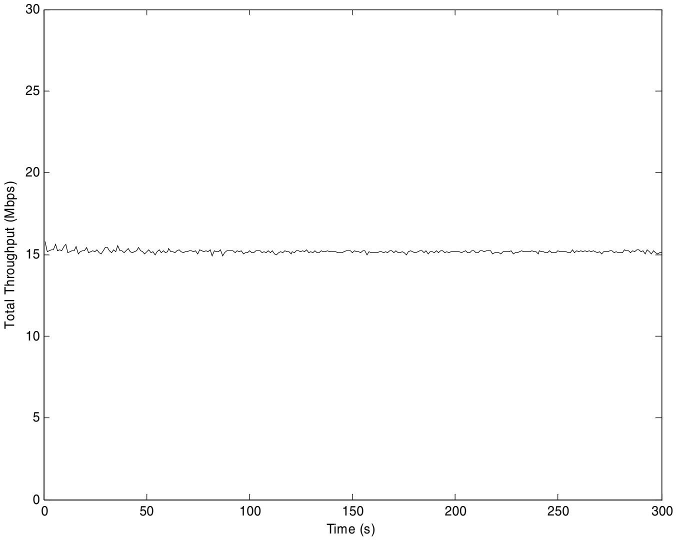


Fig. 5. Instant Throughput of the proposed scheme (Mbps)

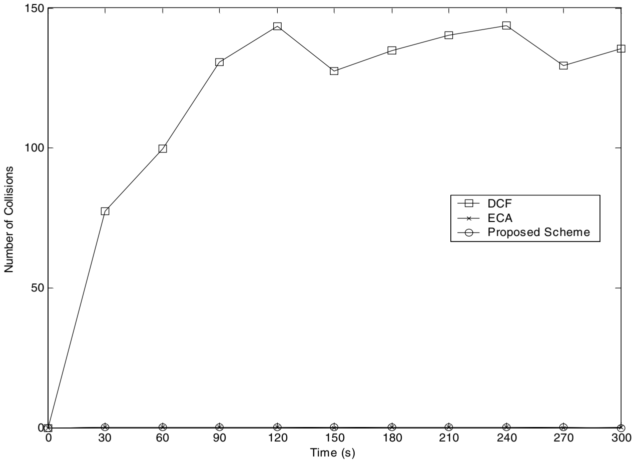


Fig. 6. Number of Collisions

scheme with the ECA, and shows that the ECA has less frequent collisions but each time with a relatively higher number of collisions than the proposed scheme, whereas the proposed scheme has more frequent collisions but each time with a small number of collisions.

At this moment, readers may wonder that since the ECA's number of collisions is not very bad, why it has a worse throughput performance than the proposed scheme. The answer is that the proposed scheme reduces a lot of empty slots than the ECA.

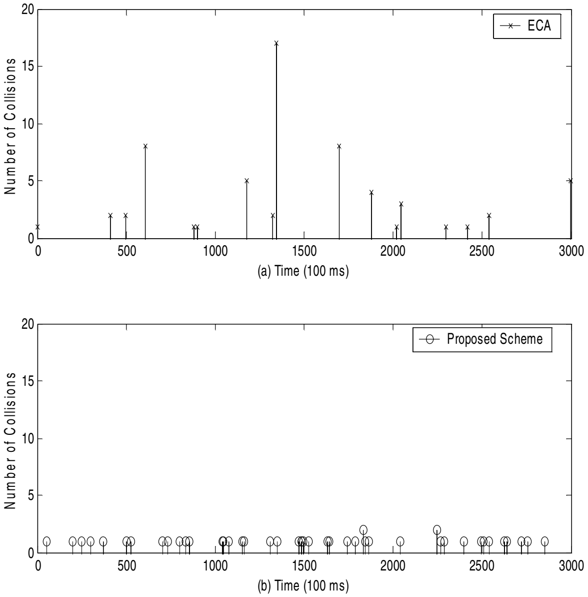


Fig. 7. Number of Collisions

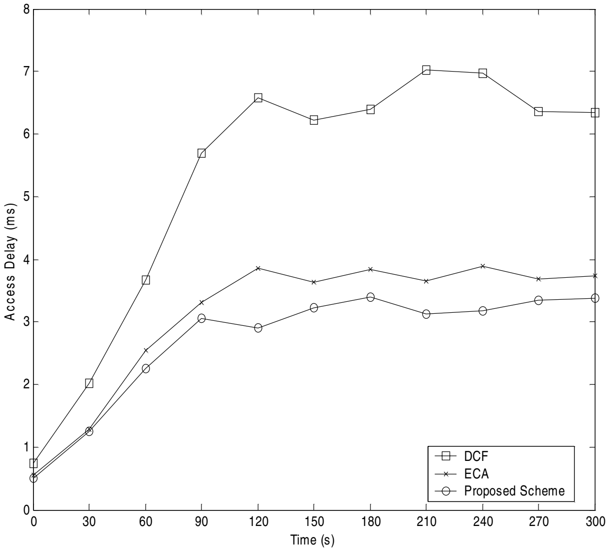


Fig. 8. Access Delay

Fig. 8 shows the access delays of all three schemes, where the access delay is defined as the time interval between when a station is ready to transmit a frame (the frame is in head of the waiting queue) and when the frame has been successfully transmitted. As illustrated by the figure, both the proposed scheme and the ECA outperform the DCF, and the proposed scheme is the best.

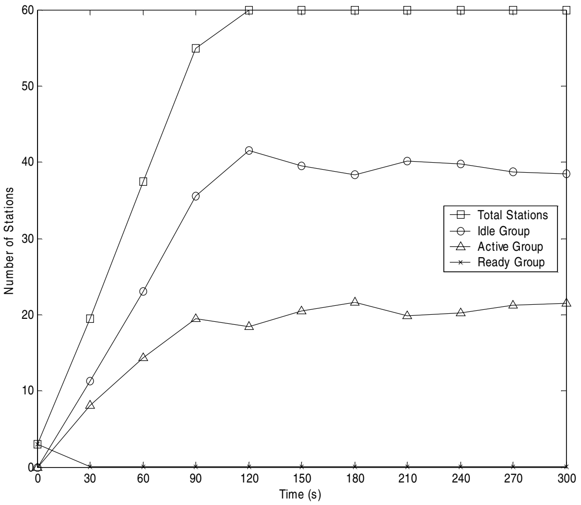


Fig. 9. Grouping

Fig. 9 shows the number of stations in the active, ready, and idle groups of the proposed scheme in this simulation setup. It shows that the number of stations in the ready state is pretty small, but not zero. The numbers of stations in the active group and the idle groups are relatively large.

In fact, we also perform simulations in many other situations such as when the number of stations in the ready group is large too, and we also have similar conclusions. We omit the results due to the limited space.

4 Conclusion

This paper proposed a novel contention-based protocol based on the concept of grouping stations as well as announcing the backoff counters in advance. Simulation studies were performed to compare with the new protocol with the DCF and ECA, and our results show that the proposed scheme is the best in terms of throughput and delay. The proposed scheme outperforms the DCF and ECA because: 1) announcing the backoff counters in advance avoids possible collisions among active stations; and 2) grouping stations enable one to estimate the number of ready stations, thus determining the appropriate window size to reduce collisions among ready stations.

References

1. IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification, 1999
2. F. Calì, M. Conti, and E. Gregori, "Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit," IEEE/ACM Trans. Networking, Vol. 8, No. 6, Dec. 2000, pp. 785-790.

3. F. Cali, M. Conti, and E. Gregori, “IEEE 802.11 Protocol: Design and Performance Evaluation of an Adaptive Backoff Mechanism,” IEEE J-SAC, Vol. 18, No. 19, Sep. 2000, pp. 1774-1786.
4. G. Bianchi, “Performance Analysis of the IEEE 802.11 Distributed Coordination Function,” IEEE J-SAC Vol. 18 No. 3, Mar. 2000, pp. 535-547.
5. E. Ziouva and T. Antonakopoulos, “CSMA/CA performance under high traffic conditions: throughput and delay analysis,” Computer Communications, 25 (2002), pp.313-321.
6. B. Bing and R. Subramanianb, “A novel technique for quantitative performance evaluation of wireless LANs,” Computer Communications, Vol. 21, No. 9, July 1998, pp.833-838.
7. K. C. Huang and K.-C. Chen, "Interference analysis of nonpersistent CSMA with hidden terminals in multicell wireless data networks", in Proc. IEEE PIMRC, Toronto, Canada, Sept. 1995, pp.907-911.
8. H. S. Chhaya and S. Gupta, "Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol", Wireless Networks, vol. 3, pp. 217-234, 1997.
9. Y. C. Tay and K. C. Chua, “A Capacity Analysis for the IEEE 802.11 MAC Protocol,” Wireless Networks 7, 2001, pp. 159-171.
10. Y. Xiao and J. Rosdahl, “Throughput and Delay Limits of IEEE 802.11,” IEEE Communications Letters, Vol. 6, No. 8, Aug. 2002, pp. 355-357.
11. Y. Xiao and J. Rosdahl, “Performance Analysis and Enhancement for the Current and Future IEEE 802.11 MAC Protocols,” ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), Vol. 7, No. 2, Apr. 2003, pp. 6-19.
12. K. K. Leung, B. McNair, L.J. Cimini, and J.H. Winters, "Outdoor IEEE 802.11 Cellular Networks: MAC Protocol Design and Performance," Proc. of IEEE ICC 2002, New York, April 2002.
13. R. O. Baldwin, N. J. Davis IV, and S. E Midkiff, “A real-time Medium Access Control Protocol for Ad hoc Wireless Local Area Networks,” Mobile Computing and Communications Review (MC2R), Vol. 3, No. 2, pp. 20-27, Apr. 1999.
14. R. O. Baldwin, N. J. Davis IV, S. E. Midkif, and R. A. Raines, “Packetized Voice Transmission using RT-MAC, a Wireless Real-time Medium Access Control Protocol,” Mobile Computing and Communications Review (MC2R), Vol. 5, No. 3, pp. 11-25, Jul. 2001.
15. IEEE 802.11a: High-speed Physical Layer in the 5GHz Band, 1999.

Path Switching in OBS Networks[⋆]

Li Yang and George N. Rouskas

Department of Computer Science,
North Carolina State University,
Raleigh, NC, USA
{lyang, rouskas}@eos.ncsu.edu

Abstract. We investigate the concept of path switching in optical burst switched (OBS) networks and its potential to reducing the overall burst drop probability. With path switching, each source maintains a list of alternate paths to each destination, and uses information regarding the recent congestion status of the network links to rank the paths; it then transmits bursts along the least congested path. We present a suite of path switching strategies, each utilizing a different type of information regarding the link congestion status, and evaluate them using simulation. Our results demonstrate that, in general, path switching outperforms shortest path routing, and, depending on the path strategy involved, the network topology, and the traffic pattern, this improvement can be significant. We also present a new framework for the development of hybrid path switching strategies, which make routing decisions based on a weighted combination of the decisions taken by several independent path switching strategies. We present two instances of such hybrid strategies, one that assigns static weights and one that dynamically adjusts the weights based on feedback from the network.

1 Introduction

Optical burst switching (OBS) is a promising switching paradigm which aspires to provide a flexible infrastructure for carrying future Internet traffic in an effective yet practical manner. OBS transport is positioned between wavelength routing (i.e., circuit switching) and optical packet switching. All-optical circuits tend to be inefficient for traffic that has not been groomed or statistically multiplexed, whereas optical packet switching requires practical, cost-effective, and scalable implementations of optical buffering and optical header processing, which are several years away. OBS does not require buffering or packet-level parsing in the data path, and it is more efficient than circuit switching when the sustained traffic volume does not consume a full wavelength. The transmission of each burst is preceded by the transmission of a setup message whose purpose is to reserve switching resources along the path for the upcoming data burst. An OBS source node does not wait for confirmation that an end-to-end connection

[⋆] This work was supported in part by MCNC-RDI as part of the Jumpstart project.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 406–418, 2005.

has been set-up; instead it starts transmitting a data burst after a delay (referred to as "offset"), following the setup message. For a recent overview of the breadth and depth of current OBS research, the reader is referred to [1].

One of the most important issues in OBS networks is that of burst loss due to congestion, and appropriate mechanisms must be in place to manage the increased demand for resources during a period of congestion. Such mechanisms can be implemented either inside the network (at core switches) or at the edge. Contention-resolution schemes at the core of the network can be based on one of four orthogonal approaches, or a combination thereof: buffering, wavelength conversion, burst segmentation, or deflection [2, 3]. All these approaches require additional hardware or software components at each OBS switch, increasing their cost significantly; furthermore, practical implementations of some of these components require technology which may be several years from maturity.

At the network edge, one strategy that has the potential to improve burst contention, especially when wavelength conversion is unavailable or sparse, is wavelength assignment [4]. A traffic engineering approach to select paths for source routing so as to balance the traffic load across the network links was investigated in [5]. Finally, a dynamic scheme for selecting routes at the burst sources was proposed in [6]. We note that a similar technique, referred to as "end-to-end path switching" was proposed and evaluated recently for selecting one among a set of Internet paths [7]; the main finding of the study was that path switching can result in substantial improvement in packet loss.

In this paper, we undertake a comprehensive study of (end-to-end) path switching in OBS networks. The remainder of the paper is organized as follows. In Section 2 we discuss our assumptions regarding the OBS network we consider in our study. In Section 3 we describe path switching strategies each utilizing partial information about the network state to select one of a set of available paths to route bursts. In Section 4 we develop a framework for combining several path switching strategies into hybrid (or meta-) strategies which base their routing decisions on the decisions of multiple individual methods. In Section 5, we present simulation results to demonstrate the effectiveness and benefits of path switching, and we conclude the paper in Section 6.

2 The OBS Network Under Study

We will use $G = (V, E)$ to denote an OBS network. V is the set of switches, $N = |V|$, and $E = \{\ell_1, \ell_2, \ldots, \ell_M\}$ is the set of unidirectional fiber links, $M = |E|$. Each link in the network can carry burst traffic on any wavelength from a fixed set of W wavelengths, $\{\lambda_1, \lambda_2, \cdots, \lambda_W\}$. We assume that each OBS switch in the network has full wavelength conversion capabilities which are used in the case of wavelength contention. The network does not use any other contention resolution mechanism. Specifically, OBS switches do not employ any buffering, either electronic or optical, in the data path, and they do not utilize deflection routing or burst segmentation. Therefore, if a burst requires an output port at a time when all wavelengths of that port are busy transmitting other bursts, then the burst is dropped.

The OBS network employs source routing, in that the ingress switch determines the path of a burst entering the network. We assume the existence of a routing algorithm that is capable of computing a set of $k, k = 2 - 4$, alternate paths for each source-destination pair. Each source node maintains the list of paths for each possible destination, and is responsible for selecting the path over which a given burst will travel. Once the source has made a routing decision for a burst, the path is recorded in the setup message and it cannot be modified by downstream nodes (i.e., no deflection is allowed).

All source nodes use the same *path switching strategy* to make routing decisions on a per-burst basis. A path switching strategy is characterized by the metric used to rank the paths to a certain destination node. In general, the metric is designed to reflect the likelihood that a burst transmitted on a particular path will experience resource contention and be dropped before it reaches its destination. Whenever a new burst is ready for transmission, the source node selects the “best” path according to the metric used (with ties broken arbitrarily) and injects the burst into the network. We present a number of path switching strategies, and their associated metrics, in the next section.

The rank of each path maintained at a source node is updated dynamically based on information regarding the state of the network collected by the node. We assume that the control plane of the OBS network provides support for the collection and dissemination of information required by the path switching strategies. For instance, this information may be part of the feedback the source receives from the signaling protocol regarding the success or failure of each burst transmission [8]. Alternatively, the OBS switches may collect information and statistics regarding the (long-term) congestion status of their links, and use a link-state protocol to disseminate this information to the rest of the network. Since signaling and state dissemination protocols are required for a variety of network functions, the additional overhead due to the path switching strategies we propose in this paper is expected to be only moderate.

3 Pure Path Switching Strategies

A path switching strategy uses information about the current state of the OBS network to select one of a small number of routing paths for transmitting burst traffic between a source-destination pair. There are several different pieces of information that could be used to describe the congestion level in the network (for instance, link utilization, end-to-end path burst drop rate, etc.); and there are several ways in which this information can be combined into a metric to rank paths. It is unknown which types of information or what metrics perform best for path switching in terms of burst drop probability. In this section we present a suite of *pure* path switching strategies, i.e., strategies which use a single path selection method.

3.1 Weighted Bottleneck Link Utilization (WBLU) Strategy

The WBLU strategy ranks the candidate paths using information on link utilization. The motivation behind this strategy is to reduce or prevent contention by using paths with less utilized links. Consider a (directional) link ℓ of the OBS network, let $Succ(\ell, t)$ denote the set of bursts that have successfully traversed link ℓ until time t, and let T_i denote the length of burst i. The utilization $U(\ell, t)$ of link ℓ at time t is defined as:

$$U(\ell, t) \quad = \quad \frac{\sum_{i \in Succ(\ell,t)} T_i}{Wt} \tag{1}$$

where W is the number of available wavelengths; at time $t = 0$, we assume that the utilization $U(\ell, 0) = 0$ for all links ℓ.

Consider now a source-destination pair (s, d), and let $\{\pi_z, z = 1, \cdots, m\}$ be the set of m candidate paths for transmitting bursts from node s to node d. Let $\{\ell_k, k = 1, \cdots, |\pi_z|\}$ be the set of links composing path π_z which has length (in number of hops) $|\pi_z|$. At time t, the WBLU strategy routes bursts from s to d along the path $\pi_{z^\star(t)}$ whose index $z^\star(t)$ is obtained using the following metric:

$$z^\star(t) \quad = \quad \arg \max_{1 \le z \le m} \frac{1 - \max_{1 \le k \le |\pi_z|} U(\ell_k, t)}{|\pi_z|} \tag{2}$$

The numerator in the above expression is the available capacity of the bottleneck link in a given path π_z. Therefore, the WBLU strategy routes bursts along the path with the highest ratio of available bottleneck link capacity to path length. By taking the number of hops into account as in expression (2), we ensure that if the bottleneck link utilization is similar for two paths, then the shortest path is selected for routing; the longer path is preferred only if the utilization of its bottleneck link is significantly lower than that of the shorter one. We note that a similar metric for ranking paths was used in [9] as part of a routing and wavelength assignment algorithm for wavelength routed networks.

3.2 Weighted Link Congestion (WLC) Strategy

The objective of the WLC strategy is to route bursts along the path that is most likely to lead to a successful transmission. To this end, the source uses information on link congestion along each path to infer the burst drop rate of the path. This strategy assumes the existence of a link-state protocol that disseminates information on link congestion.

Let $N_{succ}(\ell, t)$ (respectively, $N_{drop}(\ell, t)$) denote the number of bursts that have been successfully transmitted along (respectively, dropped at) link ℓ up to time t. We define the congestion level $c(\ell)$ of link ℓ at time t as the fraction of bursts that have been dropped at the link:

$$c(\ell, t) \quad = \quad \frac{N_{drop}(\ell, t)}{N_{drop}(\ell, t) + N_{succ}(\ell, t)} \tag{3}$$

We assume that at time $t = 0$, the congestion $c(\ell, 0) = 0$ for all links ℓ.

Let π_z be a candidate path for routing bursts between a source-destination pair (s, d), consisting of links $\ell_1, \cdots, \ell_{|\pi_z|}$. Assuming that link drop probabilities are independent, at time t the probability that a burst will be dropped along this path can be calculated as:

$$b(\pi_z, t) \quad = \quad 1 - \prod_{1 \le i \le |\pi_z|} (1 - c(\ell_i, t)) \tag{4}$$

The weighted link congestion (WLC) strategy routes bursts from s to d along the path $\pi_{z^\star(t)}$ whose index $z^\star(t)$ is obtained using the following metric:

$$z^\star(t) \quad = \quad \arg \max_{1 \le z \le m} \frac{1 - b(\pi_z, t)}{|\pi_z|} \tag{5}$$

As in expression (2), this metric takes the number of hops of each path into account, in order to ensure that longer paths are preferred over shorter ones only when they offer a substantial improvement in drop probability.

3.3 End-to-End Path Priority-Based (EPP) Strategy

The EPP strategy is similar in spirit to the WLC strategy in that it also attempts to route bursts along paths with low drop probability. However, rather than relying on information on individual link congestion levels to infer the burst drop probability, this strategy requires the source to directly measure this probability from feedback messages it receives from the network regarding the status of each burst transmission.

Consider the source-destination pair (s, d), and let π_z be one of the m candidate paths for this pair as before. Let $N_z(t)$ denote the total number of bursts that have been transmitted (successfully or unsuccessfully) from s to d on path π_z up to time t. The EPP strategy assigns a priority $prio(\pi_z, t)$ to path π_z at time t which is updated each time a new burst is transmitted on this path, and is recursively defined as:

$$prio(\pi_z, t) \quad = \quad \begin{cases} 1.0, & t = 0 \\ \frac{prio(\pi_z, t-1) \times N_z(t-1) + 1}{N_z(t-1)+1}, & \text{burst transm. success at time } t \\ \frac{prio(\pi_z, t-1) \times N_z(t-1)}{N_z(t-1)+1}, & \text{burst transm. failed at time } t \end{cases} \tag{6}$$

$N_z(t)$ is also updated as: $N_z(t) = N_z(t-1) + 1$ each time a new burst is transmitted on path π_z, with $N(0) = 0$. In the above expressions, the time index t refers to the time the source receives feedback from the network regarding the outcome (success or failure) of the most recent burst transmission along path π_z; similarly, index $t-1$ refers to the time feedback was received regarding the immediately previous burst transmission over the same path. The priority of a path remains unchanged in the interval $[t-1, t)$. The priority of a path in expression (6) is simply the fraction of bursts that have been successfully transmitted along this path up to time t; hence, the range of path priorities is in (0,1).

At time t, the EPP strategy routes bursts from s to d along the path $\pi_{z^\star(t)}$ whose index $z^\star(t)$ is obtained using the following metric:

$$z^\star = \begin{cases} z, & prio(\pi_z, t) - prio(\pi_x, t) > \varDelta \ \forall \ x \neq z \\ \arg\max_{1 \leq z \leq m} \frac{prio(\pi_z, t)}{|\pi_z|}, & \text{otherwise} \end{cases} \tag{7}$$

The threshold $\varDelta$ reflects the degree of confidence in the selection of a given path for routing paths. If we are sufficiently confident that a path is better than others in terms of burst drop probability, then the selection is based solely on path priorities. Otherwise, we discount the priority of each path by its length, and we select a path based on the discounted priorities.

4 Hybrid Path Switching Strategies

Each pure path switching strategy uses only one piece of information in reaching a decision, and this information provides only a limited "view" of the network state. In this section, we focus on hybrid strategies which, at each burst transmission instant, combine the decisions of several pure strategies into an overall decision in the hope of improving the accuracy of the path selection process and improve the overall burst drop probability. In general, a hybrid strategy emulates a set of pure strategies which run independently of each other "on the side." The motivation for this approach is to combine the different partial "views" of the network state in a way that improves the performance.

The principles underlying the hybrid path switching strategies are based on ideas from the domain of machine learning [10, 11]. Specifically, it has been shown [10] that the *ensemble* decision reached by a set of voters is more accurate than the decision of any individual voter, provided that each voter reaches a decision in a manner that is largely independent of other voters. In the context of path switching in an OBS network, a pure path switching strategy corresponds to a voter, and the selection of a path corresponds to a (routing) decision. A strategy is "correct" if transmitting the burst over the path selected by the strategy is successful, and it is "wrong" if the burst is dropped along the path before it reaches its destination. We can think of the overall burst drop probability of a strategy as its "error rate," i.e., the fraction of time the method is incorrect in successfully selecting a path for a burst. Obviously, the drop probability *overestimates* the real error rate of the strategy, since the fact that a burst is dropped along a given path does not necessarily imply that the burst would have been successful had another path been chosen. We expect that making routing decisions by considering several different views simultaneously will lead to better performance in terms of burst drop probability.

In the remainder of this section, we consider a single pair (s, d); our observations apply similarly to all other source-destination pairs. The source s maintains $m > 1$ candidate paths, $\pi_1, \cdots, \pi_m$, for routing bursts to d. For ease of presentation we will drop any references to the pair (s, d).

Let us assume that there are n pure path switching strategies available, $S_1, S_2, \cdots, S_n$. A strategy S_i takes as input some information regarding the network state, and produces a probability distribution $p_i^{(z)}$ over the indices of the candidate paths. The probability $p_i^{(z)}, z = 1, \cdots, m$, represents the degree of confidence that strategy S_i has in selecting candidate path π_z for routing the burst traffic. Obviously, we have: $p_i^{(1)} + p_i^{(2)} + \cdots + p_i^{(m)} = 1$.

A hybrid strategy H assigns a probability distribution q_i over the n pure path switching strategies $S_1, \cdots, S_n$. The probability q_i represents the degree of confidence of the hybrid strategy H that strategy S_i is correct in its selection of a path. Again, we have that: $q_1 + q_2 + \cdots + q_n = 1$. Then, the expected confidence of the hybrid strategy in selecting candidate path π_z is:

$$E_z = \sum_{i=1}^{n} q_i p_i^{(z)} \qquad z = 1, \cdots, m \tag{8}$$

Therefore, the decision of the hybrid strategy H is to route bursts along the path $\pi_z^\star$ with the maximum expected confidence, i.e., the one with index $z^\star$:

$$z^\star = \arg \max_{1 \le z \le m} E_z \tag{9}$$

4.1 Majority Binary Voting (MBV) Strategy

Majority binary voting (MBV) is the simplest hybrid strategy. Let us assume that there are n pure strategies available, $S_i, \cdots, S_n$, where n is odd. Each strategy S_i makes a binary decision for each of the m candidate paths: whether to select it for routing bursts or not. Formally, the probability distribution $p_i^{(z)}$ returned by each strategy S_i is as follows:

$$p_i^{(z)} = \begin{cases} 1, \ S_i \text{ selects path } \pi_z \\ 0, \text{ otherwise} \end{cases} \qquad i = 1, \cdots, n, \ z = 1, \cdots, m \tag{10}$$

The path selected by the hybrid MBV strategy is the one with the most number of votes. This strategy assumes a uniform distribution q_i over the set $\{S_i\}$.

4.2 Weighted Non-binary Voting (WNV) Strategy

MBV restricts the pure path switching strategies to vote for a single path. Non-binary voting allows each pure strategy S_i to assign a degree of confidence to each candidate path π_z through a probability distribution $p_i^{(z)}$. One approach to obtaining the probability distribution is to normalize the values $v_i^{(z)}$ (e.g., priority, congestion level, etc.) assigned to the various paths by strategy S_i:

$$p_i^{(z)} = \frac{v_i^{(z)}}{\sum_{l=1,\cdots,m} v_i^{(l)}} \qquad i = 1, \cdots, n, \ z = 1, \cdots, m \tag{11}$$

The weighted non-binary voting (WNV) strategy further assigns a probability distribution q_i over the set of pure strategies $\{S_i\}$, and reaches a decision using expressions (8) and (9). The main motivation for using a non-uniform distribution q_i is the fact that each pure strategy results in a different burst drop probability; furthermore, the relative performance of the various pure strategies depends on system parameters such as the network topology, the traffic load and pattern, etc. In general, the performance of the hybrid strategy depends strongly on the choice of weights, with the best performance achieved when the weights reflect the relative error rate of the pure strategies.

4.3 Dynamic Weighted Non-binary Voting (DWNV) Strategy

Under WNV, the probability distribution q_i over the set of pure strategies $\{S_i\}$ remains fixed at all times. One problem with such an approach is the difficulty in appropriately selecting the weights q_i. Instead, it would be desirable to have a method for dynamically adjusting the probability distribution q_i in real time in a way that minimizes the overall burst drop probability; in this case, the probability distribution q_i would also converge to the optimal one. The dynamic weighted non-binary voting (DWNV) strategy is designed to achieve this objective.

Let $q(t) = (q_1(t), \cdots, q_n(t))$ be the probability distribution at time t, and let $B(t, q(t))$ be the burst drop probability of the hybrid strategy at time t when the current distribution is $q(t)$. Our objective is to obtain the distribution $q(t+1)$ at time $t+1$ such that the burst drop probability is minimized (the time indices refer to the times a burst is ready to be transmitted). In other words, we need to select the distribution $q^\star(t+1)$ such that: $q^\star(t+1) = \arg\min_{q(t+1)} B(t+1, q(t+1))$. However, it is not possible to solve the above optimization problem directly. We therefore employ a heuristic to dynamically update the q-distribution. We assume that the confidence $c_i(t)$ in the decision of a strategy S_i is reversely proportional to its burst drop probability $b_i(t)$ at time t:

$$c_i(t) \quad = \quad \frac{1}{b_i(t) + \epsilon} \qquad i = 1, \cdots, n \tag{12}$$

where ϵ is a smoothing value to avoid division by zero when $b_i = 0$. Based on the confidence c_i of choosing strategy S_i, we compute the new weight q_i as:

$$q_i(t+1) \quad = \quad \frac{c_i(t)}{\sum_{l=1,\cdots,n} c_l(t)} \qquad i = 1, \cdots, n \tag{13}$$

The computation of each of the expressions (13) warrants further discussion. The overall burst drop probability $B(t, q(t))$ of the hybrid policy is calculated at the source node using feedback from the network. However, it is not possible for the source node to calculate directly the burst drop probability $b_i(t)$ of each pure strategy S_i as required by (13). To see why a direct calculation of $b_i(t)$ is not possible, consider what happens if the hybrid strategy adopts a decision that is different than the decision of some pure strategy S_i. In this case, the feedback received by the source provides information regarding the decision made by the hybrid policy but no information regarding the decision made by S_i; hence, the source has no way of knowing with certainty whether the burst transmission would have been successful had it used the path selected by S_i instead.

To overcome this difficulty, we use the following approach to compute the burst drop probability $b_i(t)$ for a pure strategy S_i whose decision at time t does not coincide with the decision of the hybrid strategy. Let π be the path chosen by S_i, and let $prio(\pi, t)$ be the priority of (burst drop probability along) this path; this priority is computed in the course of the operation of the hybrid policy as in expression (6). Then, we use $prio(\pi_t)$ to update the drop probability of strategy S_i, by making the approximation that the outcome of routing a burst over path π at time t will be failure with probability $1 - prio(\pi, t)$ and success with probability $prio(\pi, t)$. Therefore, the burst drop probability for any pure strategy S_i whose decision at time t is to use path π, is updated as follows:

$$b_i(t+1) = \begin{cases} 0, & t = 0 \\ \frac{b_i(t) \times N}{N+1}, & \text{hybrid strategy chose } \pi \ \wedge \ \text{burst success} \\ \frac{b_i(t) \times N+1}{N+1}, & \text{hybrid strategy chose } \pi \ \wedge \ \text{burst dropped} \\ \frac{b_i(t) \times N+(1-prio(\pi,t))}{N+1}, & \text{hybrid strategy did not choose path } \pi \end{cases}$$

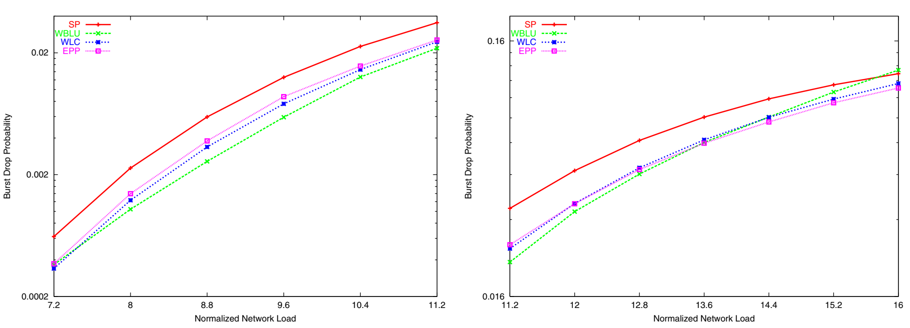
5 Numerical Results

In this section, we use simulation to investigate the performance benefits of path switching in OBS networks. We use the method of batch means to estimate the burst drop probability, with each simulation run lasting until 6×10^5 bursts have been transmitted in the entire network. We have also obtained 95% confidence intervals; however, they are so narrow that we omit them from the figures. We used two 16-node networks: a 4×4 torus network, and a 16-node network based on the 14-node NSF network. All the figures plot the burst drop probability against the "normalized network load" ρ_W, which is obtained by dividing the total load offered to the network by the number W of wavelengths: $\rho_W = \sum_{ij} \rho_{ij}/W$.

Let us first investigate the performance improvement that is possible with pure path switching over shortest path routing. We assume that each source has to select among $m = 2$ candidate paths to each destination; these are the two shortest link-disjoint paths in the network for the given source-destination pair. We compare four routing schemes: (1) Shortest-path (SP) routing, (2) WBLU path switching, (3) WLC path switching, and (4) EPP path switching.

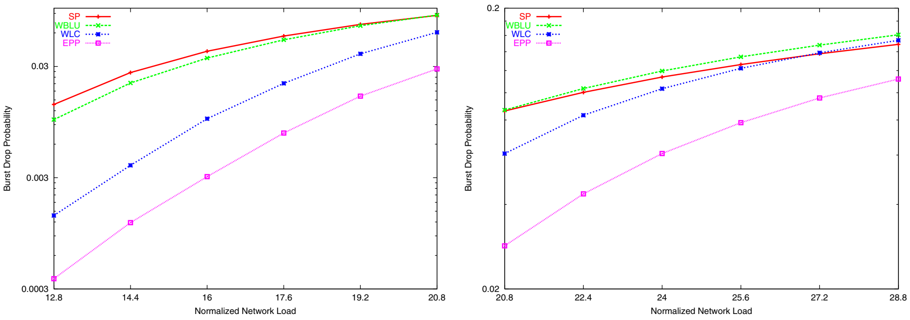
Figures 1(a)-(b) plot the burst drop probability of the above four routing schemes for the NSF network with uniform traffic. Figure 1(a) (respectively, Figure 1(b)) plots the burst drop probability for low (respectively, high) loads. As we can see, all three path switching strategies perform consistently better than SP routing throughout the load range considered in the figures; the only exception is at very high loads, where the high burst drop probability is due to a saturated network. This result demonstrates the benefits of path switching.

Another important observation is that none of the three path switching strategies is a clear winner over the entire range of loads shown. In general, WBLU performs the best at low loads, EPP is the best strategy at high loads, while the burst drop probability of WLC is between the values of the other two



(a) (b)

Fig. 1. Burst drop prob., NSF network, uniform traffic (a) Low load (b) High load

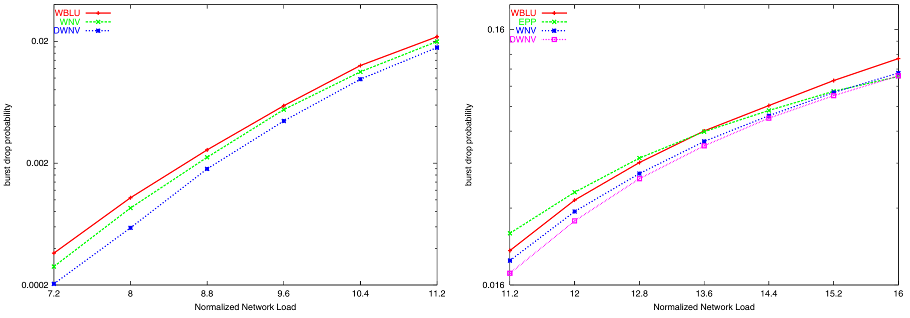


(a) (b)

Fig. 2. Burst drop prob., Torus network, uniform traffic (a) Low load (b) High load

strategies. At low network loads, most links have low utilization, and avoiding the few highly utilized (bottleneck) links can significantly improve the burst drop probability. Since the WBLU strategy takes account the bottleneck link utilization in determining the burst path, it performs well at low loads. At high loads, the EPP strategy outperforms the WBLU and WLC strategies. This behavior can be explained by the manner in which the three strategies update their path decisions. Under EPP, path priorities are updated immediately upon the receipt of feedback messages from the network, whereas the WBLU and WLC strategies update their routing decisions periodically. The period of update for WBLU and WLC is independent of the network load. With the EPP strategy, however, as the load increases, the rate of feedback from the network increases accordingly, providing a more accurate view of the network state and resulting in better routing decisions.

The performance of the four routing methods for the Torus network and uniform traffic is shown in Figures 2(a)-(b). The WLC and EPP strategies perform consistently better than SP routing, and in fact the burst drop probability of EPP is significantly lower than that of both WLC and SP across the whole range



(a) (b)

Fig. 3. Burst drop prob., NSF network, uniform traffic (a) Low load (b) High load

from low to high loads. The WBLU strategy, on the other hand, is only slightly better than SP at low loads, and slightly worse than SP at high loads. This result is due to the fact that WBLU makes a routing decision based only on the utilization of the bottleneck link. In a symmetric topology such as the Torus, WBLU leads to routing oscillations which tend to hurt the overall performance. In our experiments, we have observed that the oscillations persist throughout the simulation, and that they become worse as the offered load increases. In the asymmetric NSF network, on the other hand, we have observed that the routing decision of WBLU oscillates at first, but it later settles down to a fixed path. The only exception is at very high loads when the bottleneck links are saturated, in which case WBLU keeps oscillating among the candidate paths; this is reflected in Figure 1(b) for a load of 16, when WBLU performs worse than SP routing.

We now consider the WNV and DWNV hybrid strategies. Each hybrid strategy utilizes four routing strategies in making its decision: SP routing and the WBLU, WLC, and EPP pure path switching strategies. In order to characterize the performance of hybrid path switching, we compare the following three routing schemes: (1) WNV path switching, which assigns static weights (q-distributions) to each of the four pure strategies. We have found that different weights perform differently for each of the two topologies. Therefore, after some experimentation, we have used the following weights: for the NSF network, all weights are equal to 1/4 (a uniform distribution), while for the Torus network, the weights are: 1/8 (SP and WBLU), 1/4 (WLC), and 1/2 (EPP). (2) DWNV path switching, in which the weights of all pure strategies are initially equal, but they are adjusted dynamically during the operation of the network as in Section 4. (2) Best pure strategy, in which bursts are sent along the path determined by the pure strategy with the best performance among the four strategies SP, WBLU, WLC, and EPP. If one pure strategy is best across some range of loads while another is best across a different range, we present both strategies.

Figures 3(a)-(b) show results for the NSF network with uniform traffic; WBLU has the best performance among the pure path switching strategies at low loads, while EPP is the best pure strategy at high loads. The hybrid WNV path switching scheme improves the burst drop probability over both pure strategies; in effect, the WNV curve tracks the best of the WBLU or EPP curves. This result confirms our intuition that taking into account several different views of the network state increases the performance. When the weight of each pure strategy is adjusted dynamically to reflect the real-time network performance, as accomplished by DWNV, the burst drop probability is further improved. Our experiments indicate that through dynamic adjustments, the weights assigned to each pure strategy by a source-destination pair are fine-tuned depending on the source-destination pair and the traffic pattern. This tuning procedure can be viewed as a "dynamic optimization" process that allows the hybrid DWNV strategy to achieve a final set of weights that is near-optimal in the sense of minimizing the burst drop probability.

6 Concluding Remarks

We considered the problem of multipath routing in OBS networks and we developed a suite of path switching strategies, each utilizing one type of information regarding the network state to select one of a set of paths to route a given burst. We presented a probabilistic framework for hybrid path switching strategies which make routing decisions by taking into account the decisions of multiple pure strategies. We can summarize our results as follows. **(1)** Pure path switching strategies can reduce the burst drop probability compared to shortest path routing. **(2)** The performance improvement depends on the congestion information utilized by the strategy, the network topology, and the load; in many cases, the improvement over shortest path routing can be dramatic. **(3)** Hybrid strategies can be used to further improve the performance. However, if one pure strategy clearly outperforms all others, then a hybrid strategy may not provide any improvement. In this case, it is best to simply use the most successful pure strategy instead. **(4)** The performance is optimized when the weights assigned to the pure strategies by a hybrid strategy can be appropriately selected. Otherwise, a hybrid strategy that dynamically adjusts the weights performs best.

References

1. Chen, Y., Qiao, C., Yu, X.: Optical burst switching: A new area in optical networking research. IEEE Network **18** (2004) 16–23
2. Yao, *et al.*: A unified study of contention-resolution schemes in optical packet-switched networks. Journal of Lightwave Technology **21** (2003) 672–683
3. Vokkarane, V., *et al.*: Burst segmentation: An approach for reducing packet loss in optical burst switched networks. In: Proc. IEEE ICC. (2002) 2673–2677
4. Teng, J., Rouskas, G.N.: On wavelength assignment in optical burst switched networks. In: Proceeding of BROADNETS 2004. (2004) 24–33
5. Teng, J., Rouskas, G.N.: Routing path optimization in optical burst switched networks. In: Proc. ONDM 2005. (2005, to appear)

6. Thodime, G., *et al.*: Dynamic congestion-based load balanced routing in optical burst-switched networks. In: Proc. Globecom. Volume 5. (2003) 2628–2632
7. Tao, S., *et al.*: Exploring the performance benefits of end-to-end path switching. In: Proc. ACM SIGMETRICS 2004, New York, NY, USA (2004) 418–419
8. Baldine, I., Rouskas, *et al.*: `JumpStart`: A just-in-time signaling architecture for WDM burst-switched networks. IEEE Communications Magazine **40** (2002) 82–89
9. Chu, X.W., *et al.*: A dynamic RWA algorithm in a wavelength-routed all-optical network with wavelength converters. In: Proc. IEEE INFOCOM. (2003) 1795–1802
10. Hansen, L., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 993–1001
11. Dietterich, T.G.: Machine learning research: Four current directions. AI Magazine **18** (1997) 97–136

A Priority-Aware Protection Technique for Quality of Service Enabled WDM Networks

W. Fawaz[1], F. Martignon[2], K. Chen[1], and G. Pujolle[3]

[1] University of Paris 13 - L2TI Lab, 99, Avenue Jean-Baptiste Clement,
93430 Villetaneuse, France
m.fawaz@isep.fr
chen@galilee.univ-paris13.fr

[2] Dipartimento Ingegneria Gestionale e dell'Informazione,
University of Bergamo, Italy
fabio.martignon@unibg.it

[3] University of Paris 6, LIP6 Laboratory,
8 rue du Capitaine Scott, 75015, Paris, France
guy.pujolle@lip6.fr

Abstract. Connection availability is considered as a critical metric, when providing differentiated services in WDM mesh networks. Indeed, one of the major concerns of optical network operators is related to improving the availability of services provided to their highest-class clients. Achieving this objective is possible through the use of the different classical protection schemes, namely the so-called dedicated and shared protection schemes. However, the majority of the work concerning protection schemes has considered the primary connections as equally important when contending for the use of the backup resources.

As a main contribution in this paper, we therefore propose an improvement of the existing protection schemes through the introduction of relative priorities among the different primary connections contending for the access to the protection path. To evaluate numerically the benefits of the service differentiation feature introduced in our proposal, we first develop a mathematical model based on which we derive explicit expressions for the average connections availabilities that result from both the classical protection schemes and the proposed priority-aware one. Through this model, we show how the availability of the highest-class clients is improved when deploying the proposed priority-aware protection scheme.

Finally, with the same objective in mind, we develop a simulation study, where a given set of connection demands with predefined availability requirements is provisioned, using different protection strategies. Through this study, we show that the priority-aware protection strategy satisfies service-availability requirements in a cost-effective manner compared with the classical protection schemes.

1 Introduction

The revolutionary Wavelength-Division multiplexing (WDM) technology increases the transmission capacity of fiber links by several orders of magnitude.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 419–430, 2005.

As WDM keeps on evolving, fibers are witnessing a huge increase regarding their carriage capacity, which has already reached the order of terabits per second. Therefore, the failure of a network component (e.g., a fiber link, an optical cross connect, an amplifier, a transceiver, etc) can weigh heavily on optical carrier operators due to the consequent huge loss in data and revenue. To get an estimate of the different optical components failure characteristics, Table 1 presents the mean failure rates and failure repair times of various optical network components according to Bellcore (now Telecordia) [1], where Failure-In-Time (FIT) denotes the average number of failures in 10^9 hours, Tx denotes optical transmitters, Rx denotes optical receivers, and MTTR stands for Mean Time To Repair.

Table 1. Failure rates and repair times (Telecordia [1])

Metric	**Telecordia Statistics**
Equipment MTTR	2h
Cable-cut MTTR	12h
Cable-cut rate	501142 FIT/1000 sheat miles
Tx failure rate	10867 FIT
Rx failure rate	4311 FIT

Two main conclusions may be drawn based on these statistics: the frequency of failure occurrence in optical networks is not negligible; moreover, cable cut is the dominant failure scenario, compared to Tx and Rx failures, for lengths in the order of hundreds of kilometers, normally found in backbone optical networks. With the frequent occurrence of fiber cuts and the tremendous loss that a failure may cause, network survivability, together with its impact on network design, becomes a critical concern for operators who strive to keep up with the competition for broadband traffic transport. Moreover, as WDM networks migrate from ring to mesh topology, planning a survivable WDM mesh network has been the subject of extensive studies [3, 4, 5] leading to the definition of various resilience approaches. Mainly, there are two types of fault recovery mechanisms: *protection* [6] and *restoration* schemes [7]. In this paper we focus our study on protection schemes, dealing mainly with the impact these schemes have on the customer-perceived service quality which is an emerging topic and of special interest today. We believe that protection, a proactive procedure, is a key strategy to ensure fiber network survivability. To the best of our knowledge what still lacks in existing literature is a systematic methodology to efficiently select a cost-effective protection scheme for each connection, while satisfying its quality of service (QoS) requirements. Usually, by means of service contracts called Service Level Agreements (SLA), a client subscribes to optical network services from the optical operator with a certain guaranteed QoS level. Within the SLA, Service Level Specifications (SLS) [8] quantify the quality of service provided to the customer. A certain number of SLSs indicate the reliability constraints needed by the subscribed service. Reliability parameters presented in the literature include mainly service availability, and restoration time. Our interest will be directed to service availability since the problem of how connection availability is affected by network failures is currently attracting more research interest.

As a first main contribution in this paper, we propose an extension for the so-called shared protection scheme contributing to the design of new quality of service-aware protection schemes. In order to gauge the benefits of our proposal, the impact of such an approach on the customer-perceived service availability needs to be studied and to be compared with the classical protection approaches. Moreover, to assess the efficiency of the proposed scheme in comparison to the classical protection schemes, we need to evaluate the cost in terms of resources (i.e., number of wavelengths needed for instance) induced from the deployment of both the priority-aware scheme and the classical schemes in a real network. Therefore, we first present a mathematical model for both the classical shared-protection schemes and the proposed priority-aware scheme. We derive explicit analytic expressions for the average availability resulting from the deployment of such schemes. By solving these models we then evaluate numerically the benefits of the service differentiation feature introduced in our scheme. Finally, we exhibit the cost-effectiveness of our proposed approach regarding resource consumption (i.e., wavelengths) in a sample optical network topology using a simulation study. In this regard, a given set of randomly generated connection demands with predefined availability requirements are routed in the network using several provisioning schemes (i.e., using unprotected, dedicated, shared, and priority-aware shared protection schemes). The performances of these provisioning schemes are compared in terms of resources needed in the network, and in terms of the connections availability satisfaction rate. Our results show that the proposed protection approach achieves a high satisfaction rate while greatly economising resource usage. The paper is structured as follows: in Section II we revise the related works presented in the litterature, pointing out our position relative to these works; in Section III we propose and describe the priority-aware shared protection scheme; in Section IV we introduce a mathematical model to evaluate the impact of the protection schemes analyzed in this paper on the connection availability; in Section V we present numerical results based on the mathematical study to evaluate the benefits of the service differentiation feature introduced in our scheme. In section VI, the simulation study is developed with the corresponding results. Finally, Section VII concludes this paper and proposes future issues.

2 Priority-Aware Shared-Protection Scheme

This Section introduces the proposed protection scheme that extends the existing shared-protection schemes through the introduction of relative priorities among different primary connections contending for the backup paths. Let us consider N working paths $(w_i, i = 1, \ldots, N)$ with the same source and destination sharing M backup paths $(b_i, i = 1, \ldots, M)$, i.e. an M:N protections scheme, as depicted in Figure 1. Both work paths and backup paths can be in failure. When a failure occurs, the repair process is started.

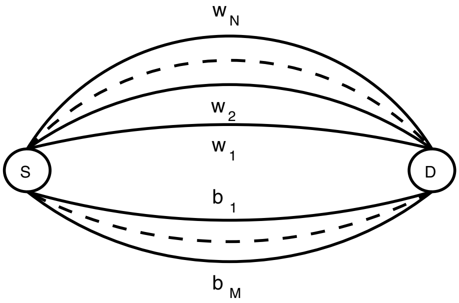


Fig. 1. N working paths sharing M backup paths between a source node S and a destination node D

In the classical shared-protection scheme, when several subsequent failures happen in the network, all connections are considered of equal importance when contending for backup resources. As such, the first failed connection gains access to the backup path. On the other hand, in our proposed scheme these connections are divided into K sets of reliability classes, $C_1, \ldots, C_K$, with N_i connections belonging to class C_i for $i = 1$ to K, and $\sum_{i=1}^{K} N_i = N$. Connections belonging to class C_1 have the highest priority, while those belonging to C_K have the lowest priority. When the working path of a connection t belonging to class C_i breaks down, the first available backup path, if any, is assigned to protect connection t and restoration is ensured by switching t to the backup path. Meanwhile, repair actions are performed on the primary path to restore it to be as good as new. Once repairing the primary path is achieved, the restored connection is switched back to its primary path. On the contrary, if at the moment t fails all the backup paths are already occupied protecting other connections, a check is made to verify the existence of protected connections belonging to classes of lower priority than t, i.e. to classes comprised between $i+1$ and K. If several such connections exist, the one having the lowest priority is immediately preempted by connection t. The preempted connection thus becomes unavailable, waiting for a backup path to be freed or for its working path to be repaired. Finally, if neither of the two above situations is verified, connection t becomes unavailable.

3 The Mathematical Model

In this Section, we present a mathematical model for both the classical 1:N shared protection scheme and the corresponding priority-aware extension discussed previously. Solving this model, we derive explicit expressions for the average availability of a connection resulting from deploying the aforementioned protection strategies. It is important to note that the dedicated protection case can be viewed as a special case of the shared protection scheme with N=1. As we are interested in the *availability* of a connection, we need to define it first. The availability of a connection is defined as the probability that such connection is “up” at any given time [9], and can be expressed as the proportion of time the connection is up during its entire service. If a connection is carried by a single unprotected path, its availability is equal to the path availability. The availability of a protected connection is determined by both the primary and the backup paths. In other words, a protected connection t is said to be *available* when either no failure affects its primary path or it is recovered by the backup path in case of failure along the primary path. Connection t becomes *unavailable* in the following two cases:

- one failure occurs on the primary path of t and a second failure occurs on its backup path;
- if t shares the backup path with connection t', then t will be unavailable if both t and t' fail but the shared backup path is taken by t'. In the priority-aware scheme, this happens if t' has higher priority than t.

3.1 Model Definition and Resolution for the Classic Shared-Protection Scheme

Let us consider N working paths that share the same backup path, i.e. a 1:N shared-protection scheme. Let $\lambda_i, i = 1, ..., N+1$ be the mean failure rate of the *i-th* path and μ_i be the mean recovery rate of the *i-th* path; $\frac{1}{\lambda_i}$ and $\frac{1}{\mu_i}$ hence represent the Mean Time To Failure and Mean Time To Repair of the *i-th* path, respectively. Based on the above assumptions, all the path failures are statistically independent, and interfailure and repair times are exponentially distributed. To gain insight into the behavior of the system and according to existing literature [10, 11], we will consider a case of special interest in which all the paths (working as well as backup ones) have identical failure and recovery rates, i.e. $\lambda_i = \lambda$ and $\mu_i = \mu, \forall i = 1, \dots, N+1$. Let us define $\rho = \frac{\lambda}{\mu}$. We have here a classical problem of reliability, with 1 redundant unit for N working units. Here a unit is an optical path. The steady-state availability A_i of a single path i, viz. the limiting $(\tau \to \infty)$ probability of finding the path successfully operating at time t, can be calculated as follows:

$$A_i = \frac{MTTF}{MTTF + MTTR} = \frac{1/\lambda}{1/\lambda + 1/\mu} = \frac{1}{1+\rho} \tag{1}$$

$\overline{A_i} = 1 - A_i$ represents the unavailability of path i.

Let $F(\tau)$ be the number of failed paths at time t. Because of the assumptions, $F(\tau); \tau \geq 0$ forms a continuous and stationary Markov process, with $F(0) = 0$. Let $p(n)$ be the steady state probability that $F(\tau) = n$ in stationary regime. The transition diagram is given in Fig. 2.

After some classical calculus we can express the steady state probability $p(n)$ of the Markov chain as follows [10, 12]:

$$p(n) = C_{N+1}^{n} \overline{A}^{n} A^{N+1-n} = \frac{(N+1)!}{n!(N+1-n)!} \frac{\rho^n}{(1+\rho)^{N+1}} \tag{2}$$

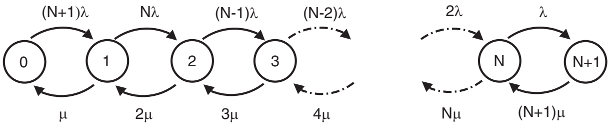


Fig. 2. Transition Diagram

where C_{N+1}^{n} represents the number of all combinations of n failed paths out of $N+1$, and A is given by equation (1). In other words, the number of failed paths follows a binomial distribution with parameters $N+1$ and $\overline{A}$.

Note that $p(n)$ represents the proportion of time in which there are n failures in the network. When the total number of path failures n is greater than or equal to one, we can distinguish two cases:

1. the backup path is among the failed paths and the remaining $n-1$ connections cannot be restored;
2. all the n failed paths are primary paths, and as such, only one connection is restored by the backup path while the remaining $n-1$ are not.

Therefore, under such conditions there will always be exactly $n-1$ unavailable connections. For $n \geq 2$ at least one connection will be unavailable, while when the number of failures n is equal to 1, there will be no unavailable connections. From this classical result, we are now interested in calculating the average unavailability of a specific connection t among the N shared-protected ones. The average unavailability of t is the proportion of time such connection is unavailable for all possible numbers of failures n, $2 \leq n \leq N+1$. Let us define $Y(n)$ the event of t being unavailable under state n. The probability of having our reference connection t unavailable when there are n failed paths is equal to $p(n)P(Y(n))$. As $p(n)$ has already been calculated in equation (2), what remains is to calculate $P(Y(n))$. To do so, we have to consider all the events that may lead to the connection t becoming unavailable under state n. These events are the following: $W(n)$: both the primary path of connection t and the backup path are failed; $Z(n)$: connection t's primary path is failed but the backup path is available.

Building on this information and according to the theorem of total probability, $P(Y(n))$ can be calculated as follows

$$P(Y(n)) = P(Y(n)|W(n))P(W(n)) + P(Y(n)|Z(n))P(Z(n)) \tag{3}$$

where $P(Y(n)|W(n))$ and $P(Y(n)|Z(n))$ are, respectively, the conditional probabilities of having our reference connection t unavailable, given that events $W(n)$ and $Z(n)$ occurred. $P(Y(n)|W(n)) = 1$ as the backup path in this case is failed and no restoration is possible; $P(Y(n)|Z(n)) = \frac{n-1}{n}$ as only one of the n primary paths under failure in this case can be restored.

The probability of the event $W(n)$ is:

$$P(W(n)) = \frac{C_{N-1}^{n-2}}{C_{N+1}^{n}} = \frac{n(n-1)}{N(N+1)} \tag{4}$$

where the numerator indicates all possible combinations where the primary path of connection t and the backup path are among the failures. The denominator indicates all possible combinations of n failed paths out of $N+1$.

The probability of the event $Z(n)$ is:

$$P(Z(n)) = \frac{C_{N-1}^{n-1}}{C_{N+1}^{n}} = \frac{n(N+1-n)}{N(N+1)} \tag{5}$$

where the numerator indicates all possible combinations where the primary path of the connection t is among the failures while the backup is not.

Then, based on the above equations, the probability P(Y(n)) that the observed connection t is unavailable under state n is equal to:

$$P(Y(n)) = \frac{n-1}{N},\ 2 \leq n \leq N+1 \tag{6}$$

It can be seen that this equation is also valid for the case $n = 1$, for which $P(Y(n)) = 0$, since in this case all connections will be available, as stated before. Based on the theorem of total probability, the unavailability of a connection in the case of 1:N protection is given by the following formula:

$$U(N, \lambda, \mu) = \sum_{n=2}^{N+1} p(n) \cdot P(Y(n)) = \sum_{n=2}^{N+1} p(n) \cdot \frac{n-1}{N} \tag{7}$$

and, substituting the expression (1) for $p(n)$ we obtain:

$$U(N, \lambda, \mu) = \frac{1}{N} \cdot \sum_{n=2}^{N+1} \frac{(n-1) \cdot C_{N+1}^{n} \cdot \rho^n}{(1+\rho)^{N+1}} \tag{8}$$

The average availability for a connection is simply equal to $1 - U(N, \lambda, \mu)$.

3.2 Model Definition and Resolution for the Priority-Aware Scheme

Let us consider the priority-aware shared-protection system proposed in Section III, where N connections are divided into two sets of reliability classes, C_1 and C_2, with N_1 and N_2 connections belonging to class C_1 and C_2, respectively, and $N_1 + N_2 = N$. Connections of class C_1 have higher priority than connections belonging to C_2. In the following we derive the analytic expressions for the availability for each connection according to its priority class. We will begin by considering higher-priority connections. First of all, the N_1 connections having the highest priority can preempt instantaneously all the other connections belonging to the lower-priority class in the utilization of the backup path. Consequently, the analysis of the proposed scheme with regard to the high-priority connections is equivalent to the study of a classic 1:N_1 shared-protection scheme. Therefore, we can derive straightforwardly the average unavailability U_1 of high-priority class connections based on equation (8) by simply substituting N with N_1.

$$U_1(N_1, \lambda, \mu) = \frac{1}{N_1} \cdot \sum_{n=2}^{N_1+1} \frac{(n-1) \cdot C_{N_1+1}^{n} \cdot \rho^n}{(1+\rho)^{N_1+1}} \tag{9}$$

When a low-priority connection fails, it becomes unavailable if any of the following mutually exclusive conditions is verified:

1. the protection path has already failed;
2. the protection path is up but there is at least one high-priority connection among the failures;
3. the protection path is up, no high-priority connections are among failures, there is however another low-priority connection occupying the protection path.

Let E_i be the event of having condition i verified, $i = 1, 2, 3$. Therefore, to study the unavailability U_2 of a low-priority connection, we consider the process $Q(\tau)$ whose general state is a triplet (n_1, n_2, b), where n_1 and n_2 indicate, respectively, the number of failed high and low-priority connections at time τ, and b is a flag set to 1 if the backup path is down and to 0 if it is up. $Q(\tau)$ is a continuous and stationary Markov process, with a limiting probability for each state given by

$$P(n_1, n_2, b) = P(n_1)P(n_2)P(b) \tag{10}$$

where $P(n_1)$, the probability of having n_1 failed high-priority connections and $P(n_2)$, the probability of having n_2 failed low-priority connections, are respectively equal to:

$$P(n_1) = C_{N_1}^{n_1} \overline{A}^{n_1} A^{N_1 - n_1} \tag{11}$$

$$P(n_2) = C_{N_2}^{n_2} \overline{A}^{n_2} A^{N_2 - n_2} \tag{12}$$

and A is given by equation (1). $P(b)$ is the probability of having b backup path failures. In other words, when $b = 0$, there is no failure affecting the backup path, whereas if $b = 1$ the backup path is down. The expression of $P(b)$ is:

$$P(b) = \overline{A}^b A^{1-b} \tag{13}$$

The events $(E_i, i = 1, 2, 3)$, leading to the unavailability of a low-priority connection, are verified according to the values of n_1, n_2 and b. So, $b = 1$ leads to E_1, meaning that the protection path has failed; on the other hand, $b = 0$ and $n_1 \geq 1$ lead to event E_2; finally, $b = 0$, $n_1 = 0$ and $n_2 \geq 2$ produce event E_3. Under state (n_1, n_2, b), a specific low-priority connection t is unavailable when it fails *and* one of the events $E_1 - E_3$ is produced. Based on this observation, U_2 is given by:

$$U_2 = \sum_{\forall (n_1, n_2, b)} P(t \text{ fails in state } (n_1, n_2, b)) \cdot P(n_1, n_2, b) \cdot P(E_1 \cup E_2 \cup E_3) \tag{14}$$

where:

$$P(t \text{ fails in state } (n_1, n_2, b)) = \frac{C_{N_2-1}^{n_2-1}}{C_{N_2}^{n_2}} \tag{15}$$

and $P(E_1 \cup E_2 \cup E_3)$ can be obtained with classical manipulations. It follows that U_2 is equal to:

$$U_2 = \sum_{i=2}^{N_2+1} C_{N_2-1}^{i-2} \overline{A}^i A^{N_2-i+1} + \sum_{i=1}^{N_2} C_{N_2-1}^{i-1} \overline{A}^i A^{N_2-i+1} \cdot (1 - A^{N_1}) + \\ + \sum_{i=2}^{N_2} C_{N_2-1}^{i-1} \overline{A}^i A^{N_2-i+1} \cdot A^{N_1} \cdot \frac{(i-1)}{i} \tag{16}$$

4 Numerical Results

In this Section we gauge the benefits the proposed priority-aware protection scheme through numerical results induced from the previous mathematical study. For the sake of simplicity, we consider a scenario consisting of 3 primary connections sharing one backup path.We first consider a priority-aware protection scheme, with one high-priority and two low-priority primary connections.The availability of each class is calculated for different connections' lengths based on equations (9) and (14), and is reported in Figure 3. Then, a classical shared protection scheme is applied to this scenario, and the availability of a connection is evaluated using equation (8). The corresponding results are reported again in Figure 3 for comparison purposes. It is important to state that the Mean Time To Repair ($\frac{1}{\mu}$) of all the paths is considered equal to 12 hours (see Table I).

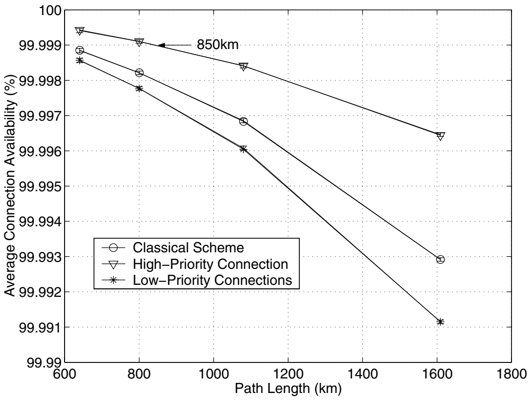


Fig. 3. Average availability for the classical and the priority-aware 1:3 shared-protection scheme

Based on Figure 3 we can observe that the high-priority connection protected using the priority-aware scheme is more available than the connections protected by the classical shared scheme. The observed availability results can be interpreted from a Quality of Service level perspective using the following reasoning. According to [8] a Platinum client requests an availability of 99.999% (i.e. at most 5 minutes of unavailability per year), whereas a Gold client requires an availability of 99.99% per year. With regard to this QoS terminology, the high and the low-priority classes can be mapped into Platinum and Gold QoS levels [8] or to lower QoS classes according to the connection's length.

In fact, as shown in Figure 3, the availability of the high-priority connection drops below 99.999% when the connection length exceeds 850km, while in the classical scheme this target availability is never achieved. This proves that by deploying the proposed scheme, Platinum connections provisioning becomes possible in the network even for long communications which are encountered typically in backbone optical networks. Moreover, the QoS level of the Gold connections is still maintained.

5 Simulation Study

For illustration purposes and following the guidelines presented in [13], in our simulation we consider the network topology of Figure 4; availability of fibers is a pre-assigned value which could be 99.8%, 99.9%, 99.95%, or 99.995% according to their length. A traffic matrix of connection requests among all node pairs (i.e., Total number of connections = 24×23 connections) is randomly generated. The availability requirements of the connection requests are uniformly distributed between two classes: 99.9%, or 99.99%, which are referred to as Silver and Gold classes respectively. The traffic matrix is routed in the network according to different provisioning schemes which adopt distinct protection strategies. The availability for each provisioned connection is then calculated and compared to its required availability. Based on this, the Availability Satisfaction Rate resulting from each provisioning scheme for both Gold (ASR_G) and Silver (ASR_S)

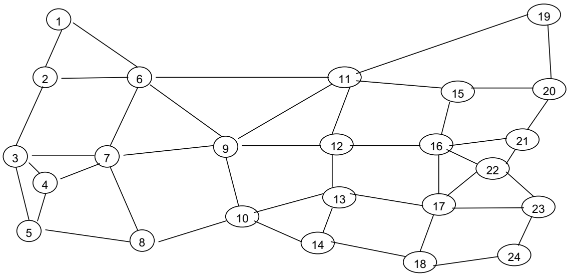


Fig. 4. A sample network topology

Table 2. Results from Four Provisioning Schemes

Scheme	ASR_G	ASR_S	W	W_{Total}
Scheme I	5%	20%	94	3352
Scheme II	100%	100%	180	7961
Scheme III	94%	100%	150	6182
Scheme IV	100%	100 %	150	6182

connections is computed and reported in Table 2 . Moreover in Table 2, the performance of the different provisioning schemes is stipulated in terms of the number of wavelength channels needed (W), and the total number of wavelength links (W_{Total}). W denotes the number of wavelength channels on the most congested fiber. On the other hand, W_{Total} denotes the total number of consumed wavelengths in the whole network.

We compare the performance of four different provisioning schemes:

Scheme I (Without Protection) where all connections are routed using a simple Dijkstra algorithm applied to the hop number without any protection, and without any connection-availability consideration; *Scheme II (Dedicated Protection)* where all connections are provisioned with dedicated-path protection (i.e.,1:1 protection); *Scheme III (Classical Shared-Path Protection)* where all connections are provisioned with the classical shared-path protection; and *Scheme IV (Priority-Aware Shared Protection)* where all connections are provisioned according to the proposed priority-aware shared-path protection.

From Table 2, one can observe that Scheme I consumes the least amount of resources compared with the other schemes. But in Scheme I, only 5% of Gold and 20% of Silver connections can meet their required availabilities. This is because the primary path in Scheme I is calculated according to the minimum number of hops but it may not be reliable enough. By deploying a dedicated protection as in Scheme II, the Gold and Silver connection Availability Satisfaction rates (ASR_G, ASR_S) reach 100%; however, a large amount of resources is consumed. By providing a classical shared protection scheme as in Scheme III, an optimization of resource usage is achieved while realizing high ASRs but the Availability Satisfaction Rate of Gold connections drops below 100%. Finally, when deploying the priority-aware protection scheme proposed in this paper (Scheme IV), the Availability Satisfaction Rates for both Gold and Silver connections (ASR_G, ASR_S) attain 100% while optimizing resource usage.

6 Conclusions

In this paper we have proposed an improvement of the existing shared protection schemes through the introduction of relative priorities among the different primary connections contending for the access to the protection path. We presented a detailed mathematical model for both the classical shared-protection schemes and for the proposed priority-aware scheme. We derived explicit analytic expressions for the average availability resulting from the deployment of such schemes. Through this study, it has been proven that service differentiation is better achieved through the use of our proposed protection scheme.

Finally, we developed a simulation study where it has been shown that the proposed scheme achieves high Availability Satisfaction Rates while realizing cost-effectiveness in terms of resource usage in the network.

References

1. Jing Zhang and B.Mukherjee. A Review of Fault Management in WDM Mesh Networks: Basic Concepts and Research Challenges. In *IEEE Network*, pages 41–48 vol.18(2), March-April 2004.
2. R.Ramaswami. Optical Fiber Communication: From Transmission to Networking. In *IEEE Communications Magazine*, pages 138–147 vol.40(5), May 2002.
3. S.Ramamurthy, L.Sahasrabuddhe, and B.Mukherjee. Survivable WDM Mesh Networks. In *Journal of Lightwave Technology*, pages 870–883, vol. 21(4), April 2003.
4. G.Mohan, S.R.Murthy, and A.K.Somani. Efficient Algorithms for Routing Dependable Connections in WDM Optical Networks. In *IEEE/ACM Transactions on Networking*, pages 553–566 vol.9, Oct. 2001.
5. G.Ellinas, A.Hailemariam, and T.E.Stern. Protection Cycles in Mesh WDM Networks. In *IEEE Journal on Selected Areas in Communications*, pages 1924–1937 vol.18, October 2001.
6. S.Ramamurthy and B.Mukherjee. Survivable WDM Mesh Networks, Part I – Protection. In *Proceedings of INFOCOM'99*, pages 744–751, March 1999.
7. S.Ramamurthy and B.Mukherjee. Survivable WDM Mesh Networks, Part II – Restoration. In *Proceedings of IEEE International Conference on Communications (ICC '99)*, pages 2023–2030, June 1999.
8. W.Fawaz, B.Daheb, O.Audouin, B.Berde, M.Vigoureux, M.Du-Pond, and G.Pujolle. Service Level Agreement and Provisioning in Optical Networks. In *IEEE Communications Magazine*, June 2004.
9. M.To and P.Neusy. Unavailability Analysis of Long-Haul Networks. In *IEEE Journal on Selected Areas in Communications*, pages 100–109 vol.12(1), 1994.
10. J.E.Angus. On Computing mtbf for a k-out-of-n:g Repairable System. In *IEEE Transactions on Reliability*, volume Vol 37(3), pages 312–313, August 1988.
11. D.Lee, L.Libman, and A.Orda. Path Protection and Blocking Probability Minimization in Optical Networks. In *Proceedings of INFOCOM'04*, 7-11 March 2004.
12. D.Mitra. Stochastic Theory of a Fluid Model of Producers and Consumers Coupled by a Buffer. In *Advances in Applied Probability*, pages 646–676 vol.20, 1988.
13. H.Zang J.Zhang, K.Zhu and B.Mukherjee. A New Provisioning Framework to Provide Availability-Guaranteed Service in WDM Mesh Networks. In *Proceedings of ICC 2003*, pages 1484–1488, May 2003.

Wavelength Rerouting in Survivable WDM Networks

Yingyu Wan and Weifa Liang

Department of Computer Science,
The Australian National University,
Canberra, ACT 0200, Australia

Abstract. One limitation of all-optical WDM networks is the wavelength continuity constraint imposed by all-optical cross-connect switches that requires the same wavelength be used on all the links along a path. With random arrivals and departures of connection requests, it happens quite often that a new request has to be blocked due to the fact that there are not enough available resources (e.g. wavelength) to accommodate the request. Wavelength rerouting, a viable and cost-effective method, which rearranges the wavelengths on certain existing routes to free a wavelength continuous route for the new request, has been proposed to improve the blocking probability. In this paper, we study a wavelength rerouting problem in survivable WDM networks as follows. Given a connection request, the problem is to find two link-disjoint paths from the source node to the destination node with an objective to minimize the number of existing routes that have to be wavelength-rerouted. We show that the problem is NP-hard if different wavelengths are assigned to the link-disjoint paths. Otherwise, a polynomial time algorithm is proposed.

1 Introduction

A Wavelength Division Multiplexing (WDM) network consists of optical wavelength routing nodes interconnected by point-to-point fiber links in an arbitrary topology. On each fiber link, the fiber bandwidth is divided into multiple frequency bands (wavelengths) so that several connection requests can be realized over it at the same time, as long as each uses a different wavelength. There are two types of WDM networks, one allows wavelength conversion at its nodes and the other does not. In a network without wavelength conversion, the realization of a connection request is subject to the *wavelength continuity constraint* that the same wavelength is used on all the links in the route for the request. This constraint often reduces the wavelength utilization because a non-wavelength-continuous route cannot be used even if it is available. This is especially severe in a network with random arrivals and departures of requests. Although wavelength conversion can potentially allow the network to accommodate more requests, it is expected that the wavelength converters at nodes are expensive in the near future. Hence, most existing work assumes that no wavelength conversion in the network is allowed. This assumption will be adopted by this paper as well.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 431–442, 2005.

To alleviate the inefficiency brought by the wavelength continuity constraint, a viable and cost-effective method called *rerouting* has been proposed, which is described as follows. Whenever a new request arrives, if there is no wavelength-continuous route for it, rearrange a certain number of existing routes to free a wavelength for the request. There are two ways to rearrange an existing route. One is "fully rearranging", which is to find a new route with another wavelength to replace the old route. This is also referred to as *nonblocking rearrangement.* Another is "partially rearranging", which keeps the original route but reassigns a different wavelength to the links in the route. This latter one is referred to as *wavelength rerouting.* Examples of nonblocking rearrangement and wavelength rerouting can be found in [1, 2, 3] and [4, 5, 6, 7, 8] respectively. While rerouting can be used to improve the bandwidth utilization, transmission on each rerouted route must be temporarily shut down to prevent data from being lost during the rerouting processing. This causes a low or even zero throughput on those being rerouted traffic. The throughput loss is particularly prominent in all-optical networks wherein each routing path is expected to carry gigabits of data flow per second, and hence even a tiny period of outage on a single route will cause significant data loss. Thus, to minimize this disruptions (i.e., the number of rerouted routes) is of paramount importance for rerouting in all-optical WDM networks. It is a general belief that nonblocking rearrangement is much harder than wavelength rerouting because in the former not only a new route for a connection request needs to be built but also another available wavelength needs to be assigned to each of the links in the new route. Despite the fact that nonblocking rearrangement may improve the blocking probability significantly, it leads to a much longer disruption than expected. In particular in an arbitrary topology network, it is very difficult to make a nonblocking rearrangement for a new request with the minimum disruption. In reality, most known works on nonblocking rearrangement are carried out on special topology networks like rings and tori [2]. On the other hand, the survivability of routing is a critical design problem for high-speed WDM networks. To protect a mission-critical connection from a single link failure, a common solution is to find a pair of link-disjoint paths from a source node to a destination node. One of the paths serves as the *active path* (or **AP**) which will be used for the data transfer actually. The other serves as the *backup path* (or **BP**) once the active path is disconnected due to a link failure on it. When a mission-critical request arrives, if there are not enough wavelengths in the network to find the AP and BP pair for the request, it is possible (sometimes necessary) to reroute a certain number of existing traffic to free a wavelength for the request.

Lee and Li [5] first introduced the wavelength rerouting concept by studying the unicast routing problem with the objective to minimize the disruption incurred due to wavelength rearranging. For an undirected WDM network of n nodes, m physical links and w wavelengths on each link, they proposed a wavelength rerouting scheme called *Parallel Move-To-Vacant Wavelength-Retuning (MTV-WR)*, which has the following advantages. First, it facilitates control because the old and new routes of rerouted traffic share the same switching nodes.

Second, it reduces the calculation because only the wavelengths on the links of existing routes need to be changed. Third, it significantly reduces the disruption period. An algorithm for implementing the MTV-WR scheme has also been proposed, which takes $O(n^3 w + n^2 w^2)$ time per unicast request [5]. Mohan and Murthy [6] later provided an $O(n^2 w)$ time improved algorithm for the problem. Caprara et al [7] studied the unicast routing problem in a directed WDM network by showing that the problem is not only NP-hard but also hard to approximate. Instead, they proposed an approximation algorithm. Wan and Liang recently [8] studied the wavelength rerouting for multicast requests in both undirected and directed networks. The two link-disjoint paths problem in a weighted graph with the objective to minimize the sum of the lengths (costs) of the two paths was well studied and can be solved in polynomial time [9]. If the objective is to minimize the length of either the longer one or the shorter one of the two paths, the problem was shown to be NP-hard [10, 11].

In this paper, our focus will be on wavelength rerouting in an arbitrary topology WDM network for link-disjoint paths routing. In particular, given a connection request, the problem is to find two link-disjoint paths for it such that the number of existing routes that need to be rerouted is minimized. Notice that the problem that we study here is different from those previous works [9, 10, 11, 12, 13, 14] that use different cost metrics. We show that the problem of concern is NP-hard if different wavelengths need to be assigned to the two paths. Otherwise, a polynomial time algorithm is devised, which takes $O(m^5 \log n)$ time, where m and n are the number of links and nodes in the network.

The rest of this paper is organized as follows. In Section 2 we define the network model and provide an overview of the wavelength rerouting scheme for link-disjoint paths routing. In Section 3 we provide an NP-hard proof of the problem. In Sections 4 we propose a polynomial algorithm.

2 Preliminaries

Network Model: An undirected WDM network $M = (N, L)$ without wavelength conversion is considered, where N is the set of communication nodes and L the set of communication links. The bandwidth of each link is divided into a set of w wavelengths $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_w\}$. It is assumed that each node has sufficient optical transmitters and receivers so that no connection request will be blocked due to lack of such resources. A connection request is denoted by a pair (s, t), where s is the source node and t is the destination node. At the time a request arrives, on a given link $e \in L$ there may be some wavelengths unavailable due to that they have been occupied by the other existing routes. Denote by Λ_e ($\subseteq \Lambda$) the set of available wavelengths on link e. Let $\mathbf{P}$ be the set of existing routes (*lightpaths*). Depending on the wavelengths used by the lightpaths, $\mathbf{P}$ can be partitioned into w disjoint subsets $\mathbf{P}_1, \mathbf{P}_2, \cdots, \mathbf{P}_w$, where the links in each path in $\mathbf{P}_i$ is assigned wavelength λ_i, $1 \leq i \leq w$. If two paths share at least one link, the paths are said to *link-intersect* with each other, otherwise they are *link-disjoint.* Obviously, any two lightpaths in set $\mathbf{P}_i$ are pairwise link-disjoint.

```
Wavelength Rerouting Scheme for Link Disjoint Paths
1. For each wavelength $\lambda_i \in \Lambda$ do
1.1  For each lightpath $P \in \mathbf{P}_i$, if there is another available wavelength $\lambda_j \in \Lambda$ $(i \neq j)$
     on each link in $P$, i.e., $\forall e \in E(P), \lambda_j \in \Lambda_e$, then $P$ can be assigned wavelength $\lambda_j$,
     and $P$ is said to be *tunable*. Otherwise, $P$ is said to be *untunable*. Let $\mathbf{P}'_i$ be the
     subset of $\mathbf{P}_i$ containing all the tunable lightpaths, and $\mathbf{P}''_i = \mathbf{P}_i - \mathbf{P}'_i$.
2. Find a pair $\{AP_i, BP_j\}$ of lightpaths between $s$ and $t$ with minimum cost such that
   (i) $AP_i$ and $BP_j$ are link-disjoint;
   (ii) wavelength $\lambda_i$ $(\lambda_j)$ is assigned to each link in $AP_i$ $(BP_j)$;
   (iii) $AP_i$ $(BP_j)$ does not link-intersect with any lightpath in $\mathbf{P}''_i$ $(\mathbf{P}''_j)$;
   (iv) the cost is defined as $|\{P \in \mathbf{P}'_i : P \cap AP_i \neq \emptyset\} \cup \{P \in \mathbf{P}'_j : P \cap BP_j \neq \emptyset\}|$, where
        $P \cap AP_i \neq \emptyset$ or $P \cap BP_j \neq \emptyset$ mean that $P$ link-intersects with $AP_i$ or $BP_j$.
3. If no such a pair exists, then the request $(s,t)$ cannot be supported by the
   wavelength rerouting scheme and will be rejected. Otherwise, the request $(s,t)$
   can be realized by rerouting a certain number of existing traffic.
```

Fig. 1. Overview of the wavelength rerouting scheme

For the sake of convenience, denote by $E(P)$ and $V(P)$ the sets of links and nodes in a path P respectively in the rest of the paper.

Overview of the Wavelength Rerouting Scheme: To accommodate a new connection request (s,t) with the survivability requirement, we need to find two link-disjoint lightpaths (a link-disjoint AP and BP pair) in the current network between s and t, and the wavelength continuity constraint on both AP and BP is imposed. This can be achieved by two phases. Phase 1: find a link-disjoint AP and BP pair between s and t, using only those available wavelengths. Phase 2: find a link-disjoint AP and BP pair with rerouting some existing lightpaths if Phase 1 fails. Phase 1 is to find two link-disjoint paths using an available wavelength, which has been studied extensively [12, 13, 14]. We therefore focus on Phase 2. If Phase 1 fails, it means that there does not exist two link-disjoint paths in the network using only the available wavelengths, and a wavelength rerouting scheme is adopted, which is described in Fig. 1. The scheme first partitions each set $\mathbf{P}_i$ into two subsets, one containing tunable paths and the other containing untunable paths, then aims to find two link-disjoint paths for the new request (s,t) which link-intersect with the tunable paths only. At Step 2 of the scheme, to minimize the total cost of the two paths is equivalent to minimize the disruption for rerouting. This step can be implemented by finding a candidate solution with the minimum cost for every possible pair of λ_i and λ_j $(1 \leq i, j \leq w)$ and choosing one with the minimum cost from these candidate solutions. Depending on whether or not $i = j$, two combinatorial optimization problems arise from Step 2. (i) If $i \neq j$, a graph $G = (V, E)$ can be constructed as follows. $V = N$ and $E = E_i \cup E_j$, where $E_i = \{e \in L \mid \lambda_i \in \Lambda_e \text{ or } \exists P \in \mathbf{P}'_i, e \in E(P)\}$ and $E_j = \{e \in L \mid \lambda_j \in \Lambda_e \text{ or } \exists P \in \mathbf{P}'_j, e \in E(P)\}$.

Definition 1 (Minimum Disruption Link-Disjoint Paths 1, MDLDP1). *Given an undirected graph $G = (V, E)$ with $E = E_1 \cup E_2$, a collection $\mathcal{P}_1$ of link-disjoint paths satisfying that $\forall P \in \mathcal{P}_1, E(P) \subseteq E_1$, another collection $\mathcal{P}_2$ of* *link-disjoint paths satisfying that $\forall P \in \mathcal{P}_2, E(P) \subseteq E_2$, a source node s and a destination node t, the objective is to construct two link-disjoint paths AP and BP between s and t such that $E(AP) \subseteq E_1$, $E(BP) \subseteq E_2$ and the sum of the number of paths in $\mathcal{P}_1$ link-intersecting with AP and the number of paths in $\mathcal{P}_2$ link-intersecting with BP is minimized.*

(ii) Otherwise ($i = j$), a graph $G = (V, E)$ can be constructed as follows. $V = N$ and $E = \{e \in L \mid \lambda_i \in \varLambda_e$ or $\exists P \in \mathbf{P}'_i, e \in E(P)\}$.

Definition 2 (Minimum Disruption Link-Disjoint Paths 2, MDLDP2). *Given an undirected graph $G = (V, E)$, a collection $\mathcal{P}$ of paths which are pairwise link-disjoint, a source node s and a destination node t, the objective is to construct two link-disjoint paths AP and BP between s and t such that the number of paths in $\mathcal{P}$ that link-intersect with them is minimized.*

3 NP-Hard Proof of MDLDP1

We prove that MDLDP1 is NP-hard by a reduction from the following problem. Link Disjoint Path Problem in Graphs with Red, Green, Blue Links (LDPPRGB).

Instance: A graph $G = (V, E)$, where each link $e \in E$ is colored red, blue or green; a source node s and a destination node t.
Question: Is it possible to establish two link-disjoint paths between s and t such that one of the paths uses only the red and green links and the other uses the blue and green links?

LDPPRGB was shown to be NP-hard by a reduction from the 3SAT problem in [14]. Given an instance of LDPPRGB, a corresponding instance of MDLDP1 can be constructed as follows. Let $G(V, E)$ be the graph with $n = |V|$ nodes and $m = |E|$ links in the given LDPPRGB instance. Define $V' = V$. Let E'_1 be the set of links in E colored red or green, E'_2 the set of links in E colored blue or green, and $E' = E'_1 \cup E'_2$. For each link $e \in E$ colored red, define a *red path* consisting only of link e. For each link $e \in E$ colored blue, define a *blue path* consisting only of link e. Let $\mathcal{P}'_1$ be the set of red paths and $\mathcal{P}'_2$ the set of blue paths. Then $G' = (V', E')$, $\mathcal{P}'_1$, $\mathcal{P}'_2$ and (s, t) form an instance of MDLDP1.

If there is a feasible solution AP and BP for the MDLDP1 instance, there is also a feasible solution for the LDPPRGB instance because $E(AP) \subseteq E'_1$ and $E(BP) \subseteq E'_2$. This reduction can be implemented in polynomial time, MDLDP1 thus is NP-hard, following that LDPPRGB is NP-hard.

Theorem 1. *MDLDP1 is NP-hard.*

4 A Polynomial Time Algorithm for MDLDP2

In this section we first give the structure properties of an optimal solution for MDLDP2. We then propose a polynomial time algorithm for the problem.

4.1 Structure Properties

Let $\{AP, BP\}$ be a feasible solution for an instance of MDLDP2. Define the cost of the solution as $c(AP, BP) = |\{P \in \mathcal{P} : P$ link-intersects with AP or $BP\}|$. For the convenience of description, we assume that $\{AP, BP\}$ is laid out in a plane with s and t on the left end and right end of AP and BP respectively which is illustrated in Fig. 2(a). Each path $P_i \in \mathcal{P}$ may link-intersect with $\{AP, BP\}$ or not. If P_i link-intersects with AP, define by $A_i = P_i \cap AP = \{P^a_{i,1}, P^a_{i,2}, \cdots\}$ the set of *maximal subpaths* shared by P_i and AP, where the "maximal subpath" means that there is no other shared subpath between P_i and AP which properly includes it. Without loss of generality, let $P^a_{i,1}, P^a_{i,2}, \cdots$ be indexed in the order from left to right along AP, i.e., a subpath closer to s has a smaller index and a subpath closer to t has a larger index. $B_i = P_i \cap BP = \{P^b_{i,1}, P^b_{i,2}, \cdots\}$ can be defined similarly. If P_i link-intersects with only either AP or BP but not both of them, P_i is called *single-intersected.* Otherwise, P_i is called *double-intersected.* If P_i is single-intersected and either $|A_i| = 1$ or $|B_i| = 1$, P_i is called *trivial-intersected.* If P_i is double-intersected, define an ordered relation on $A_i \cup B_i$ based on the order those subpaths appearing in P_i. To do so, first define a direction along P_i from one endpoint to another endpoint. For every two subpaths $P_{i,j}, P_{i,k} \in A_i \cup B_i$ in P_i, if $P_{i,j}$ appears before $P_{i,k}$ in the defined direction, $P_{i,j}$ is a *predecessor* of $P_{i,k}$, denoted by $P_{i,j} \prec P_{i,k}$, and $P_{i,k}$ is a *successor* of $P_{i,j}$, denoted by $P_{i,k} \succ P_{i,j}$. Denote by $prec(P_{i,j})$ and $succ(P_{i,j})$ the immediate predecessor and immediate successor of $P_{i,j}$ respectively if they do exist. If a double-intersected path P_i satisfies two conditions: **C1:** For any $P^a_{i,j} \in A_i$, $prec(P^a_{i,j}) \in B_i$ and $succ(P^a_{i,j}) \in B_i$ if they exist. For any $P^b_{i,j} \in B_i$, $prec(P^b_{i,j}) \in A_i$ and $succ(P^b_{i,j}) \in A_i$ if they exist. **C2:** $P^a_{i,1} \prec P^a_{i,2} \prec \cdots$ and $P^b_{i,1} \prec P^b_{i,2} \prec \cdots$, or $P^a_{i,1} \succ P^a_{i,2} \succ \cdots$ and $P^b_{i,1} \succ P^b_{i,2} \succ \cdots$. Then, P_i is called *regular-intersected.* Such an example is given in Fig. 2(b). Assume that both P_i and P_j are double-intersected. If there are two subpaths P^a_{i,k_1}, P^b_{i,k_2} in P_i and two subpaths P^a_{j,l_1}, P^b_{j,l_2} in P_j such that (i) $prec(P^a_{i,k_1}) = P^b_{i,k_2}$ or $succ(P^a_{i,k_1}) = P^b_{i,k_2}$, (ii) $prec(P^a_{j,l_1}) = P^b_{j,l_2}$ or $succ(P^a_{j,l_1}) = P^b_{j,l_2}$, (iii) P^a_{i,k_1} is on the left side of P^a_{j,l_1} in AP and P^b_{j,l_2} is on the left side of P^b_{i,k_2} in BP, then P_i and P_j are called crossing with each other. Then, we show that there is an optimal solution that meets the following properties.

Property 1. Any single-intersected path is also trivial-intersected.
Property 2. Any double-intersected path is regular-intersected.
Property 3. No two double-intersected paths cross with each other.

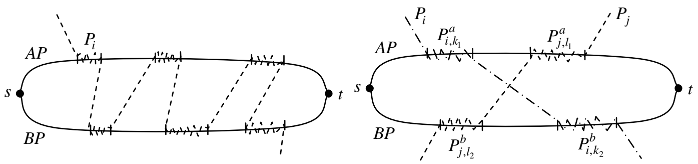


Fig. 2. (a) Regular-intersected path; and (b) P_i and P_j cross with each other

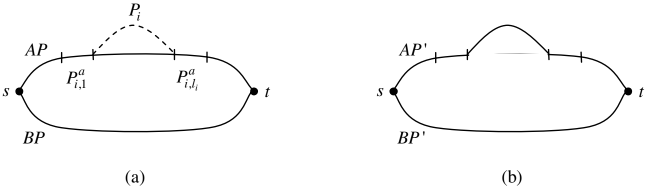


Fig. 3. Transformation to trivial-intersected path

Lemma 1. *For a feasible solution $\{AP, BP\}$, if any properties 1,2 and 3 is not met, then there is another feasible solution $\{AP', BP'\}$ such that $c(AP', BP') \leq c(AP, BP)$ and these three properties are met at the same time.*

Proof. The basic idea is to modify the solution step by step until all the properties are met. If Property 1 does not hold, without loss of generality, assume that P_i is single-intersected and $|A_i| > 1$. Let l_i be the maximum index in A_i. Let $P^a_{i,1}$ and P^a_{i,l_i} be the leftmost and rightmost subpaths of P_i in AP respectively. A new path AP' can be constructed as follows (see Fig. 3). It starts from s and traverses along AP. When it passes through $P^a_{i,1}$, it traverses along the links in P_i to P^a_{i,l_i}. It then traverses from P^a_{i,l_i} along AP to t. Because P_i is single-intersected, AP' and BP are link-disjoint. Let $BP' = BP$. A new feasible solution $\{AP', BP'\}$ is obtained. It is obvious that $c(AP', BP') \leq c(AP, BP)$ and P_i is trivial-intersected. If there are more than one single-intersected but non-trivial-intersected paths, repeat this procedure until each single-intersected path is trivial-intersected. If Property 2 does not hold, let P_i be double-intersected but non-regular-intersected. If Condition **C1** does not hold, without loss of generality, assume $P_{i,j} \in A_i$ and $succ(P_{i,j}) \in A_i$. Following the similar argument as for Property 1, $P_{i,j}$ and $succ(P_{i,j})$ can be merged into a subpath by a shortcut through the part of P_i between $P_{i,j}$ and $succ(P_{i,j})$.

Now assume that Condition **C1** but Condition **C2** holds. Consider $P^a_{i,1}$ and $P^b_{i,1}$ first. If the subpaths $prec(P^a_{i,1})$, $succ(P^a_{i,1})$, $prec(P^b_{i,1})$ and $succ(P^b_{i,1})$ exist at the same time, then $prec(P^a_{i,1}) \in B_i$, $succ(P^a_{i,1}) \in B_i$ and $prec(P^b_{i,1}) \in A_i$, $succ(P^b_{i,1}) \in A_i$. At least one of $prec(P^a_{i,1})$ and $succ(P^a_{i,1})$ is on the right of $P^b_{i,1}$ because $P^b_{i,1}$ is the leftmost subpath on BP, denoted by $P^b_{i,k}$. At least one of $prec(P^b_{i,1})$ and $succ(P^b_{i,1})$ is on the right of $P^a_{i,1}$ because $P^a_{i,1}$ is the leftmost subpath on AP, denoted by $P^a_{i,j}$. As shown in Fig. 4, a new feasible solution $\{AP', BP'\}$ can be obtained as follows. Traverse along AP from s to $P^a_{i,1}$, then traverse along P_i to $P^b_{i,k}$, finally traverse along BP from $P^b_{i,k}$ to t. Denote by AP' the resulting path. BP' can be obtained similarly. Repeat this procedure until at least one of the four subpaths $prec(P^a_{i,1})$, $succ(P^a_{i,1})$, $prec(P^b_{i,1})$ and $succ(P^b_{i,1})$ does not exist. Without loss of generality, assume $prec(P^a_{i,1})$ does not exist. If $succ(P^a_{i,1}) \neq P^b_{i,1}$, a similar transformation can be applied until $succ(P^a_{i,1}) = P^b_{i,1}$, i.e., $P^a_{i,1} \prec P^b_{i,1}$. Next consider $P^b_{i,1}$ and $P^a_{i,2}$, apply the transformation again such that $P^b_{i,1} \prec P^a_{i,2}$. Finally we have $P^a_{i,1} \prec P^b_{i,1} \prec P^a_{i,2} \prec P^b_{i,2} \prec \cdots$. Condition **C2** now holds. If Property 3 does not hold, there exist two crossing double-

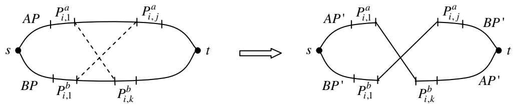


Fig. 4. Transformation to regular-intersected path

intersected paths P_i and P_j as shown in Fig. 4, by a similar transformation as for Property 2, a new feasible solution $\{AP', BP'\}$ can be found as follows. Traverse along AP from s to P^a_{i,k_1}, then along P_i to P^b_{i,k_2}, finally along BP from P^b_{i,k_2} to t. Denote by AP' the resulting path. Traverse along BP from s to P^b_{j,l_2}, then along P_j to P^a_{j,l_1}, finally along AP from P^a_{j,l_1} to t. Denote by BP' the resulting path. P_i and P_j do not cross with each other for the resulting solution $\{AP', BP'\}$.

Repeat the above transformation to the routing paths until all of the three properties in the resulting paths are met. Note that each transformation will reduce the number of subpaths by one at least, so, the procedure terminates after a certain number of transformations. Because each transformation does not increase the cost of the resulting solution, the cost of the final solution is no more than $c(AP, BP)$. □

We thus have the following theorem.

Theorem 2. *There is an optimal solution for MDLDP2 which meets Properties 1,2 and 3.*

4.2 A polynomial Algorithm

We first consider a special case of the problem where there is an optimal solution such that there is no double-intersected paths in $\mathcal{P}$ by presenting a polynomial time algorithm for it. We then propose a polynomial algorithm for the general case by removing the constraint.

No Double-Intersected Path: Assume there is an optimal solution such that there is no double-intersected paths in $\mathcal{P}$. So, there are only single-intersected or non-intersected paths in $\mathcal{P}$. If this optimal solution does not meet Property 1, then there is another optimal solution meeting Property 1, following Theorem 2. Because all the paths in $\mathcal{P}$ that are link-intersected with the optimal solution $\{AP, BP\}$ are single-intersected, the paths that are link-intersected with AP are different from the paths that are link-intersected with BP. Moreover, all the single-intersected paths are trivial-intersected. For a path P in $\mathcal{P}$ that is trivial-intersected with AP, if we traverse AP from the source node to the destination node, we will enter and leave P once and only once, which means that P can be treated like a link for AP. Based on this observation, we propose an algorithm `MDLDP-S`(G,$\mathcal{P}$,s,t), which proceeds as follows. It first constructs an auxiliary graph by compressing each path in $\mathcal{P}$ into a single directed link. Specifically, for each path $P \in \mathcal{P}$, remove all links in P from G. Add two new

```
Algorithm MDLDP-G($G$,$\mathcal{P}$,$s$,$t$)
Input: an undirected graph $G$, a set $\mathcal{P}$ of link-disjoint paths, a pair $(s, t)$.
Output: $\{AP^*, BP^*\}$, a pair of link-disjoint paths in $G$ between $s$ and $t$, such that
    the number of paths in $\mathcal{P}$ that link-intersect with $AP^*$ and $BP^*$ is minimized.
1. Construct a weighted directed auxiliary graph $G' = (V', E')$;
2. Find a shortest path in $G'$ from $s$ to $t$;
3. Construct two link-disjointed paths $\{AP^*, BP^*\}$ in $G$ between $s$ and $t$
   from the above shortest path.
```

Fig. 5. An algorithm for the general case of MDLDP

nodes *entry*(P) and *exit*(P). Add a directed link $< entry(P), exit(P) >$ from *entry*(P) to *exit*(P) and assign it weight 1. For each node v in P, add a directed link $< v, entry(P) >$ from v to *entry*(P) and a directed link $< exit(P), v >$ from *exit*(P) to v, and assign each of them weight 0. For those original undirected links in G that do not appear in any $P \in \mathcal{P}$, add them to the auxiliary graph and assign them weight 0s. Denote by $G' = (V', E')$ the resulting graph.

It then finds two link-disjoint paths $\{AP', BP'\}$ in G' with the minimum cost, using Suurballe's algorithm. An optimal solution $\{AP, BP\}$ in G is finally derived from $\{AP', BP'\}$ by a reverse procedure of the above construction. That is, for each directed link with weight 1 in AP' and BP', replace the link by the corresponding path in $\mathcal{P}$. It is easy to show that $\{AP, BP\}$ is an optimal solution for MDLDP2 in G. Obviously, the proposed algorithm takes $O(m \log n)$ time due to the facts that there are at most m paths in $\mathcal{P}$ and the total number of links in $\mathcal{P}$ is $O(m)$. Thus, there are $O(m)$ nodes and $O(m)$ links in G'. The time complexity of Suurballe's algorithm is $O(m \log n)$.

General case: Recall that Theorem 2 shows that there is an optimal solution $\{AP^*, BP^*\}$ meeting Properties 1, 2 and 3 simultaneously. Assume that $P_{i_1}, P_{i_2}, \cdots, P_{i_k}$ are all the regular-intersected paths in the order from left to right according to their appearances in $\{AP^*, BP^*\}$. For each $1 \le j \le k$, let a_j be the right node of the rightmost link P_{i_j} intersects with AP^* and b_j be the right node of the rightmost link P_{i_j} intersects with BP^*. Obviously, $a_1, a_2, \cdots, a_k$ and $b_1, b_2, \cdots, b_k$ are in the order from left to right in $\{AP^*, BP^*\}$. $\{AP^*, BP^*\}$ then can be partitioned into at most $k + 1$ pairs of link-disjoint paths by cutting at k pairs of nodes $(a_1, b_1), (a_2, b_2), \cdots, (a_k, b_k)$, and each of these pairs of nodes is referred to as a *partition-pair*. Denote by $\{AP_0^*, BP_0^*\}, \{AP_1^*, BP_1^*\}, \cdots,$ $\{AP_k^*, BP_k^*\}$ the partitions of $\{AP^*, BP^*\}$.

Let $a_0 = b_0 = s$ and $a_{k+1} = b_{k+1} = t$. For $0 \le j \le k$, each $\{AP_j^*, BP_j^*\}$ is a pair of link-disjoint paths between a_j, b_j and a_{j+1}, b_{j+1}. For each $\{AP_j^*, BP_j^*\}$, there is at most one regular-intersected path, and the other link-intersected paths are trivial-intersected. So, $\{AP_j^*, BP_j^*\}$ can be found in polynomial time, using Algorithm MDLDP-S. The optimal solution $\{AP^*, BP^*\}$ can be obtained by concatenating these $\{AP_j^*, BP_j^*\}$. Motivated by this observation, an algorithm is proposed in Fig. 5. The algorithm first constructs an auxiliary graph $G'(V', E')$, then computes a shortest path from s to t in the auxiliary graph,

```
Construction of $G'(V', E')$
1 $V' = \{s, t\}$ and $E' = \{< s, t >\}$, assign the directed link $< s, t >$ a weight, which is
    equal to the cost of the solution returned by MDLDP-S($G$,$\mathcal{P}$,$s$,$t$);
2 For each pair of nodes $\{u, v\}$, and each path $P \in \mathcal{P}$, if $u, v \in V(P)$, a partition-node
    $X_{u,v,P}$ is added to $V'$;
3 For each partition-node $X_{u,v,P}$ do
3.1 Construct a graph $G'' = (V'', E'')$: $V'' = V \cup \{t''\}$, $E'' = E$, remove all the links
      incident to $u$ or $v$ from $E''$, except those links in $P$. $E'' = E'' \cup \{(u, t''), (v, t'')\}$;
3.2 Add a new link $\{< s, X_{u,v,P} >\}$ into $E'$, and assign it a weight equal to the cost of
      the solution returned by MDLDP-S($G''$,$\mathcal{P} \setminus \{P\}$,$s$,$t''$) plus one;
4 For each partition-node $X_{u,v,P}$ do
4.1 Construct a graph $G'' = (V'', E'')$: $V'' = V \cup \{s''\}$, $E'' = E$, remove all the links
      in $P$ from $E''$. $E'' = E'' \cup \{(s'', u), (s'', v)\}$;
4.2 Add a new link $\{< X_{u,v,P}, t >\}$ into $E'$, and assign it a weight equal to the cost of
      the solution returned by MDLDP-S($G''$,$\mathcal{P} \setminus \{P\}$,$s''$,$t$);
5 For each ordered pair $< X_{u_1,v_1,P_1}, X_{u_2,v_2,P_2} >$ satisfying that $P_1 \neq P_2$, do
5.1 Construct a graph $G'' = (V'', E'')$: $V'' = V \cup \{s'', t''\}$, $E'' = E$, remove all the links
      in $P_1$ and all the links incident to $u_2$ or $v_2$ from $E''$, except those links in $P_2$.
      $E'' = E'' \cup \{(s'', u_1), (s'', v_1), (u_2, t''), (v_2, t'')\}$;
5.2 Add a new link $\{< X_{u_1,v_1,P_1}, X_{u_2,v_2,P_2} >\}$ into $E'$, and assign it a weight equal
      to the cost of the solution returned by MDLDP-S($G''$,$\mathcal{P} \setminus \{P_1, P_2\}$,$s''$,$t''$) plus one;
```

Fig. 6. Construction of the auxiliary graph in Algorithm MDLDP-G

finally constructs two link-disjoint paths in G. The construction of G' is shown in Fig. 6. In the construction, when calculate the weight of $< s, X_{u,v,P} >$, we assume that (u, v) is a partition-pair and P is the regular-intersected path. So, we remove all the links incident to u or v except those in P and remove P from $\mathcal{P}$. Other intersected paths except P are trivial-intersected. By calling MDLDP-S(G'',$\mathcal{P} \setminus \{P\}$,$s$,$t''$), we can find two link-disjoint paths between s and u, and between s and v with the minimum cost. We assign $< s, X_{u,v,P} >$ a weight that is equal to the returned cost plus one, because P is also intersected. When calculate the weight of $< X_{u,v,P}, t >$, we assume that (u, v) is a partition-pair and P is the regular-intersected path incident to u and v. We remove all the links in P and remove P from $\mathcal{P}$. All the intersected paths are trivial-intersected. By calling MDLDP-S(G'',$\mathcal{P} \setminus \{P\}$,$s''$,$t$), we find two link-disjoint paths between u and t, and between v and t. Assign the cost as the weight of $< X_{u,v,P}, t >$. The weight of $< X_{u_1,v_1,P_1}, X_{u_2,v_2,P_2} >$ can be calculated similarly. After the construction of G', we apply Dijkstra's algorithm to find a shortest path in it from s to t. Let $SP'_{s,t}$ be such a shortest path. To construct two link-disjoint paths in G between s and t using $SP'_{s,t}$, we construct a subgraph $G(SP'_{s,t})$ of G by replacing each link in $SP'_{s,t}$ with the corresponding link-disjoint paths. If a path $P \in \mathcal{P}$ is link-intersected by $G(SP'_{s,t})$, we add all the links in P into $G(SP'_{s,t})$. In the end, the number of paths in $\mathcal{P}$ appearing in $G(SP'_{s,t})$ is equal to the weight of $SP'_{s,t}$.

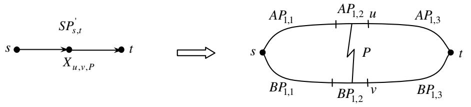


Fig. 7. Construction from $SP'_{s,t}$ to $G(SP'_{s,t})$

Lemma 2. *There are two link-disjoint paths in the subgraph $G(SP'_{s,t})$ between s and t.*

Proof. If $SP'_{s,t}$ consists of only one link $< s, t >$, then there are two link-disjoint paths found at step 1 in the construction of G'. Otherwise, there must be at least one partition-node in $SP'_{s,t}$. In the following we only consider the case where there is only one partition-node, and the case with multiple partition-nodes can be shown by similar arguments. Let $X_{u,v,P}$ be the partition-node in $SP'_{s,t}$. As shown in Fig. 7, replace the link $< s, X_{u,v,P} >$ with two paths $\{AP_{1,1} \cup AP_{1,2}, BP_{1,1} \cup BP_{1,2}\}$. And replace $< X_{u,v,P}, t >$ with the two paths $\{AP_{1,3}, BP_{1,3}\}$. $AP_{1,2}$ and $BP_{1,2}$ are the two rightmost shared subpaths of P. Because P is regular-intersected, $AP_{1,2}$ and $BP_{1,2}$ are connected by a part of P which is not link-intersected with any of $AP_{1,1}, BP_{1,1}, AP_{1,3}$ and $BP_{1,3}$, i.e., $(AP_{1,1} \cup AP_{1,2}) \cap (BP_{1,1} \cup BP_{1,2}) = \emptyset$, $AP_{1,1} \cap AP_{1,2} = \emptyset$, $BP_{1,1} \cap BP_{1,2} = \emptyset$,$(AP_{1,2} \cup AP_{1,3}) \cap (BP_{1,2} \cup BP_{1,3}) = \emptyset$, $AP_{1,2} \cap AP_{1,3} = \emptyset$, $BP_{1,2} \cap BP_{1,3} = \emptyset$. For each link e in $AP_{1,2}$, because $e \notin BP_{1,1} \cup BP_{1,2} \cup BP_{1,3}$, there is a path in $G(SP'_{s,t})$ between s and t which does not include e and e is not an st-bridge in $G(SP'_{s,t})$, where an st-bridge in $G(SP'_{s,t})$ is a link in $G(SP'_{s,t})$ such that every path in $G(SP'_{s,t})$ between s and t passes through this link. Similarly for each link e in $BP_{1,2}$,e is not an st-bridge. For each link e in $AP_{1,1}$, if $e \notin BP_{1,3}$, because $e \notin BP_{1,1} \cup BP_{1,2}$, there is a path between s and t through $BP_{1,1} \cup BP_{1,2} \cup BP_{1,3}$ not through e. Thus, e is not an st-bridge. Otherwise if $e \in BP_{1,3}$, because $AP_{1,3} \cap BP_{1,3} = \emptyset$, $e \notin AP_{1,3}$. There is a path between s and t not via e: traverse from s along $BP_{1,1}$ to $BP_{1,2}$, then traverse from $BP_{1,2}$ along P to $AP_{1,2}$, finally traverse from $AP_{1,2}$ along $AP_{1,3}$ to t. Thus e is not an st-bridge. Similarly, each link in $AP_{1,3} \cup BP_{1,1} \cup BP_{1,3}$ is not an st-bridge either. There is no st-bridge in $G(SP'_{s,t})$. The lemma follows. □

It is clear that the weight of $SP'_{s,t}$ is no more than the cost of the optimal solution for MDLDP2. Algorithm `MDLDP-G` thus is correct. The rest is to analyze the running time of the proposed algorithm. Assume there are k paths in $\mathcal{P}$ and their lengths are $l_1, l_2, \cdots, l_k$ with $l_1 + l_2 + \cdots + l_k \leq m$. There are $l_1^2 + l_2^2 + \cdots + l_k^2 \leq (l_1 + l_2 + \cdots + l_k)^2 \leq m^2$ partition-nodes in G'. The dominant step in the construction of G' is step 5, which takes $O(m^5 \log n)$-time due to that there are $O(m^2)$ nodes and $O(m^4)$ links in G'. Step 2 of algorithm `MDLDP-G` takes $O(m^4 \log n)$-time. The time complexity of algorithm `MDLDP-G` is thus $O(m^5 \log n)$.

Theorem 3. *There is a polynomial algorithm for MDLDP2 which takes $O(m^5 \log n)$ running time, where m and n are the number of links and nodes in the graph.*

Acknowledgment

It is acknowledged that the work by the authors is fully funded by a research grant No:DP0449431 by Australian Research Council under its Discovery Schemes.

References

1. A. Narula-Tam, P. J. Lin, and E. Modiano, “Efficient routing and wavelength assignment for reconfigurable WDM networks,” *IEEE Journal on Selected Areas of Communications*, vol. 20, pp. 75–88, 2002.
2. P. Saengudomlert, E. Modiano, and R. Gallager, “On-line routing and wavelength assignment for dynamic traffic in WDM ring and torus networks,” in *Proc. IEEE INFOCOM’03*, San Francisco, CA, April 2003, pp. 1805 – 1815.
3. L.-W. Chen and E. Modiano, “Efficient routing and wavelength assignment for reconfigurable WDM networks with wavelength converters,” in *Proc. IEEE INFOCOM’03*, San Francisco, CA, April 2003, pp. 1785 – 1794.
4. P. Saengudomlert, E. Modiano, and R. Gallager, “Dynamic wavelength assignment for WDM all optical tree networks,” *Allerton Conference on Communications, Control and Computing*, October 2003.
5. K. C. Lee and V. O. K. Li, “A wavelength rerouting algorithm in wide-area all-optical networks,” *J. of Lightwave Tech.*, vol. 14, pp. 1218–1229, 1996.
6. G. Mohan and C. Murthy, “A time optimal wavelength rerouting for dynamic traffic in WDM networks,” *J. of Lightwave Tech.*, vol. 17, pp. 406–417, 1999.
7. A. Caprara, G. F. Italiano, G. Mohan, A. Panconesi, and A. Srinivasan, “Wavelength rerouting in optical networks, or the venetian routing problem,” *Journal of Algorithms*, vol. 45, no. 2, pp. 93–125, 2002.
8. Y. Y. Wan and W. Liang, “Wavelength rerouting for on-line multicast in WDM networks,” in *Proc. IEEE LCN’04*, 2004.
9. J. Suurballe, “Disjoint paths in a network,” *Networks*, vol. 4, pp. 125–145, 1974.
10. C. Li, S. McCormick, and D. Simchi-Levi, “The complexity of finding two disjoint paths with min-max objective function,” *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105–115, 1990.
11. D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, “On finding disjoint paths in single and dual link cost networks,” in *Proc. IEEE INFOCOM’04*, 2004.
12. A. Sen, B. Shen, S. Bandyopadhyay, and J. Capone, “Survivability of lightwave networks - path lengths in WDM protection scheme,” *Journal of High Speed Networks*, vol. 10, no. 4, pp. 303–315, 2001.
13. S. Ramamurthy and B. Mukherjee, “Survivable WDM mesh networks part i-protection,” in *Proc. IEEE INFOCOM’99*, New York, NY, March 1999, p. 744–751.
14. R. Andersen, F. Chung, A. Sen, and G. Xue, “On disjoint path pairs with wavelength continuity constraint in WDM networks,” in *Proc. IEEE INFOCOM’04*, 2004.

Performance of Server Selection Algorithms for Content Replication Networks[*]

David Starobinski[1] and Tao Wu[2]

[1] ECE Dept., Boston University
staro@bu.edu
[2] Nokia Research Center Boston
tao.wu@nokia.com

Abstract. In this paper, we investigate the problem of optimal server selection in "content replication networks," such as peer-to-peer (P2P) and content delivery networks (CDNs). While a number of server selection policies have been proposed or implemented, understanding of the theoretical performance limits of server selection and the relative performance of existing policies remains limited. In this paper, we introduce a mathematical framework, based on the $M/G/1$ Processor Sharing queueing model, and derive closed-form expressions for the optimal server access probabilities and the optimal average delay. We also analyze the performance of two general server selection policies, referred to as EQ_DELAY and EQ_LOAD, that characterize a wide range of existing algorithms. We prove that the average delay achieved by these policies can theoretically be as much as N times larger than the optimal delay, where N is the total number of servers in the system. Furthermore, simulation results obtained using our $M/G/1$-PS workload model and the ProWGen Web workload generator show that the optimal policy can reduce the average delay of requests by as much as 30% as compared to EQ_LOAD and EQ_DELAY, in realistic scenarios. They also show that the optimal policy compares favorably to the other policies in terms of fairness and sensitivity to traffic parameters.

1 Introduction

Content replication has emerged as one of the most useful paradigms for the provision of scalable and reliable Internet services [1]. With content replication, the same data (e.g., Web pages, MP3 files, etc.) is stored at multiple geographically distant servers. Request by clients are then forwarded to one of these servers, usually the one that minimizes the delay perceived by the client. Because of its inherent scalability and fault-tolerance, content replication has become the cornerstone of most modern networking architectures, including content delivery networks (CDNs) and peer-to-peer (P2P) networks [2, 3].

One of the key issues arising with content replication is that of server selection. In most of the existing network architectures, a number of *server-selection nodes* (e.g., request redirection routers in CDNs [2] or supernodes in P2P networks [3]) are re-

[*] This work was supported in part by NSF CAREER award ANI-0132802 and DOE Early Career Principal Investigator award DE-FG02-04ER25605

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 443–454, 2005.

sponsible for aggregating incoming client requests and forwarding them to one of the servers. The main question within this context is to determine the optimal server selection policy that minimizes the overall average delay of requests in the network. We note that this problem is fundamentally different from traditional load-balancing problems, which usually assume that all the servers are co-located [4].

Obviously, because server response time is an increasing function of the load, the best solution to the server-selection problem is usually not the one that directs all the requests to a single server (e.g., the fastest), which would slow down or even crash the server [5]. Thus, a number of server selection policies have been proposed in the literature or implemented in commercial products to better balance the load over the servers. Generally speaking, these policies fall within one of the following two categories: 1) Equal load (EQ_LOAD), where the access probabilities to the servers are set so that all servers have the same utilization; round-robin, or weighted-round-robin for heterogeneous servers [4], and certain adaptive algorithms such as Least Loaded [6] and WebSeAl [7], are examples of policies following this approach. 2) Equal delay (EQ_DELAY), where the access probabilities are set so that the average delay at all the selected servers is equal or at least on the same order. The intuition behind this approach is to avoid forwarding request to very slow servers, as may happen with EQ_LOAD. SPAND [8] and the application layer anycast architecture of [5] implement variations of this approach.

In the absence of any analytical benchmark, it is unclear whether the aforementioned policies perform optimally and, if not, how well they perform compared to the optimal policy. Our goal in this paper is to address these fundamental questions and explore the strength and limitations of existing server selection policies. To this end, we model the behavior of the servers using an $M/G/1$-PS (Processor Sharing) queuing framework [9]. Using this framework, we formulate our problem as a constrained non-linear optimization problem. We show that the problem at hand has enough special structure for allowing the derivation of *closed-form* expressions for the optimal server access probabilities and the optimal average delay. Using a similar analytical procedure, we also derive expressions for the access probabilities and average delay of the EQ_LOAD and EQ_DELAY policies.

Our major theoretical findings in this paper are as follows. First, our analysis reveals that both EQ_LOAD and EQ_DELAY perform sub-optimally. Second, assuming that the system consists of a total of N servers, we prove that the average delay achieved with these policies can be as much as N times larger than the optimal average delay. Finally, although EQ_DELAY may achieve a much lower average delay than EQ_LOAD at low load, we prove that these policies perform identically at high load.

We illustrate our analytical results with an extensive number of numerical examples and simulations. Specifically, numerical results obtained using our $M/G/1$-PS workload model show that, in realistic network configurations, the optimal policy can reduce the average delay by as much as 30% as compared to EQ_LOAD and EQ_DELAY. Moreover, simulation results obtained using the ProWGen Web workload generator [10] show similar performance gain for synthetic traces exhibiting temporal locality (i.e., temporal correlation) in the popularity of files. Other simulation results show that the fairness properties of the optimal policy are comparable or even better than those of the two other policies.

The remainder of the paper is organized as follows. In Section 2, we introduce our model and notations. In Section 3, we derive the optimal server selection policy and obtain a closed-form expression for the optimal average delay. In Section 4, we analyze the EQ_DELAY and EQ_LOAD policies and compare their performance with that of the optimal policy. In Section 5, we present our simulation results and we conclude the paper in Section 6.

2 Model and Problem

2.1 Model and Notations

We consider a network consisting of M server-selection nodes and N servers. We assume that each server-selection node j generates requests following a Poisson process with rate λ_j. Further, we assume that the requests generated by different server-selection nodes are independent. Therefore, the aggregate request process is also Poisson with rate $\lambda = \sum_{i=1}^{M} \lambda_i$. Each server-selection node assigns request to server i with probability p_i, independently of other requests. Therefore, the arrival process to each server i, $i = 1, 2, \ldots, N$, is an independent Poisson process with rate $p_i\lambda$. We note that measurements of request arrivals to Web servers have been shown to match well a Poisson process, at least over small to moderate timescales [9]

We denote the service capacity of each server i by C_i. The file size distribution is arbitrary but identical at all the servers. We denote the mean file size by $\bar{x}$. The mean service rate of requests is denoted by $\mu_i = C_i/\bar{x}$. We assume that each server implements a *Processor Sharing* (PS) scheduling policy, where all the requests share the service capacity equally and continuously [11]. Processor Sharing is an idealization of the round-robin policy implemented by most existing Web servers, whereby each request is given an equal share of the service capacity [12]. Finally we assume that network delays are negligible compared to the delay that a request experienced at the server. This assumption is in line with recent measurement studies which have shown that, with over-provisioning, delays across backbones are now mostly dominated by speed-of-light delays with minimal queueing delay in the routers [13, 14].

Under the above assumptions, each server behaves as an $M/G/1$-PS queue. The average delay of a request forwarded to server i is therefore given by the following expression [11]

$$\overline{T}_i(p_i) = \frac{1}{\mu_i - p_i\lambda}. \tag{1}$$

We note that the average delay depends on the file size distribution only through its mean.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_N)'$ denote the service access probability vector. Then, for a given vector $\mathbf{p}$, the average delay of a request in the system is

$$\overline{T}(\mathbf{p}) = \sum_{i=1}^{N} p_i \overline{T}_i(p_i) = \sum_{i=1}^{N} \frac{p_i}{\mu_i - p_i\lambda}. \tag{2}$$

Note that any feasible vector $\mathbf{p}$ must satisfy several conditions. First, to be a legitimate probability vector, the coordinates of $\mathbf{p}$ must be non-negative and sum to one, that is $p_i \geq 0$ for all i and $\sum_{i=1}^{N} p_i = 1$. In addition, the coordinates of $\mathbf{p}$ must satisfy the *individual stability conditions*, i.e., $p_i < \mu_i/\lambda$, to guarantee that the arrival rate of requests is smaller than the service rate at each server i. We denote by $\mathcal{P}$ the set of vectors $\mathbf{p}$ that satisfy all the above constraints.

The set $\mathcal{P}$ is non-empty if and only if the *aggregate stability condition* $\lambda < \sum_{i=1}^{N} \mu_i$ is satisfied. To see this, define $\mu = \sum_{i=1}^{N} \mu_i$. If the aggregate stability condition holds, then the probability vector $\mathbf{p} = (\mu_1/\mu, \mu_2/\mu, \ldots, \mu_N/\mu)'$ is always feasible. On the other hand, if the aggregate stability condition does not hold, then the individual stability can never all be simultaneously satisfied. Therefore, we will assume from now and on that the aggregate stability condition always holds.

3 Derivation of the Optimal Policy

In this section, we determine the optimal vector $\mathbf{p}^*$ that minimizes the average delay expression in Eq. (2). Specifically, our problem can be formally stated as follows:

Problem 1 (OPT). Find the optimal server access probability vector

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in\mathcal{P}} \overline{T}(\mathbf{p}).$$

Problem 1 has already been the subject of studies in the literature under the general context of load sharing in queueing networks [15, 16] as well as flow assignment [17]. However, our contribution here is to provide *closed-form* expressions for the optimal server access probabilities for the specific case of $M/G/1$-PS servers. These expressions will be used to prove our main result (Theorem 4 in Section 4).

As a first step to obtain the optimal server access probabilities, we note that there exists a unique solution to Problem 1, since $\overline{T}(\mathbf{p})$ is strictly convex over $\mathcal{P}$. Next, in order to solve the constrained optimization problem, we make use of Lagrange multiplier techniques [18]. We start by defining the Lagrangian function

$$\begin{aligned} L(\mathbf{p}, l, \mathbf{m}) &= \overline{T}(\mathbf{p}) + l(\sum_{i=1}^{N} p_i - 1) - \sum_{i=1}^{N} m_i p_i \\ &= \sum_{i=1}^{N} \frac{p_i}{\mu_i - p_i\lambda} + l(\sum_{i=1}^{N} p_i - 1) - \sum_{i=1}^{N} m_i p_i, \end{aligned} \tag{3}$$

where l and $\mathbf{m} = (m_1, m_2, \cdots, m_N)$ are the so-called Lagrange multipliers. The Lagrange multiplier l enforces the equality constraint $\sum_{i=1}^{N} p_i = 1$, while the multipliers m_i enforce the inequality constraints $p_i \geq 0$. In the sequel, we denote by $I(\mathbf{p})$ the set of *inactive* servers to which requests are never forwarded, that is, $I(\mathbf{p}) = \{i \mid p_i = 0\}$.

Since $\overline{T}(\mathbf{p})$ is strictly convex and the set of constraints is convex as well, the Karush-Kuhn-Tucker (KKT) conditions stated below are both necessary and sufficient for the existence of a global minimum $\mathbf{p}^*$, assuming that the individual stability conditions are satisfied (see Proposition 3.3.4 in [18]).

Theorem 1 (KKT). Let $\mathbf{p}^*$ be the optimal solution of Problem 1. Then, there exist unique Lagrange multipliers l^* and $\mathbf{m}^* = (m_1^*, m_2^*, \cdots, m_N^*)$, such that

1. $\nabla_{\mathbf{p}} L(\mathbf{p}^*, l^*, \mathbf{m}^*) = 0$,
2. $m_i^* = 0, \quad \forall i \notin I(\mathbf{p}^*)$,
3. $m_i^* \geq 0, \quad \forall i \in I(\mathbf{p}^*)$.

Without loss of generality, we can assign the server indices so that $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_N$. We now observe that in the optimal solution, a faster server should always serve a larger fraction of requests than a slower server. Therefore, the access probabilities must satisfy the following ordering $p_1^* \geq p_2^* \geq \cdots \geq p_N^*$. As a result, the set of inactive servers $I(\mathbf{p}^*)$ must be one of the following: $\emptyset, \{N\}, \{N-1, N\}, \ldots, \{2, 3, \ldots, N\}$.

Let us denote the slowest active server in the optimal solution as N^*, that is, $I(\mathbf{p}^*) = \{N^*+1, N^*+2, \ldots, N\}$. The following theorem establishes the value of N^* and provides a closed-form expression for the optimal solution $\mathbf{p}^*$. This theorem is proven by showing that the optimal solution satisfies all the KKT conditions stated in Theorem 1 as well as the individual stability conditions. Due to space limitation, the proof is omitted here but can be found in our technical report [19].

Theorem 2. The optimal solution $\mathbf{p}^*$ to Problem 1 can be obtained as follows. Define

$$\alpha_i = \frac{\mu_i}{\lambda} - \frac{\left(\sum_{j=1}^{i} \mu_j - \lambda\right)\sqrt{\mu_i}}{\lambda \sum_{j=1}^{i} \sqrt{\mu_j}} \qquad 0 \leq i \leq N. \tag{4}$$

Then,

1. N^* is the maximum index i for which $\alpha_i > 0$.
2. $I(\mathbf{p}^*) = \{N^*+1, N^*+2, \ldots, N\}$,
3. $p_i^* = \frac{\mu_i}{\lambda} - \frac{\left[\sum_{j=1}^{N^*} \mu_j - \lambda\right]\sqrt{\mu_i}}{\lambda \sum_{j=1}^{N^*} \sqrt{\mu_j}} \quad \forall i \notin I(\mathbf{p}^*)$,
4. $p_i^* = 0 \quad \forall i \in I(\mathbf{p}^*)$.

We have developed an efficient algorithm based on Theorem 2 to determine N^* and $\mathbf{p}^*$. This algorithm requires the computation of only N expressions. It starts by assuming $I(\mathbf{p}^*) = \emptyset$, that is, server N is the slowest active server. If $\alpha_N > 0$, then the assumption is valid and the coordinates of $\mathbf{p}^*$ are set using the expressions given in Theorem 2. However, if $\alpha_N \leq 0$, then the assumption is not valid and it must be the case that $p_N^* = 0$. The algorithm proceeds by repeating the same procedure with the sets $I(\mathbf{p}^*) = \{N-i+1, N-i+2, \ldots, N\}$, $i = 1, 2, \ldots$, until an index i is found such that $\alpha_i > 0$, at which point N^* is determined. The access probabilities p_i^*, $i \notin I(\mathbf{p}^*)$, are then computed using Theorem 2. The pseudo-code for this algorithm can be found in [19].

We can now determine the average delay achieved by the optimal policy, by substituting the derived expression of $\mathbf{p}^*$ into Eq. (2):

$$\overline{T}(\mathbf{p}^*) = \sum_{i=1}^{N^*} \frac{p_i^*}{\mu_i - p_i^* \lambda} = \frac{\left(\sum_{i=1}^{N^*} \sqrt{\mu_i}\right)^2}{\lambda \left(\sum_{i=1}^{N^*} \mu_i - \lambda\right)} - \frac{N^*}{\lambda}. \tag{5}$$

4 Performance Analysis of the EQ_DELAY and EQ_LOAD Policies

In this section, we evaluate the performance of the EQ_DELAY and EQ_LOAD policies. We derive the server access probabilities of each policy. Using these results, we show that at high load, when all the servers are active, EQ_DELAY and EQ_LOAD achieve the same average delay. We also prove our main result that the average delay of EQ_DELAY and EQ_LOAD can be as much as N times larger than the optimal average delay, where N denotes the total number of servers in the system.

4.1 General Solution for EQ_DELAY

The goal of the EQ_DELAY policy is to set the probability access vector such that the average delay at all the active servers is the same and minimal. To formalize the problem, define the set

$$\mathcal{S} = \{\mathbf{p} \mid \overline{T}_i(p_i) = \overline{T}_j(p_j) \quad \forall i, j \notin I(\mathbf{p})\}. \tag{6}$$

We can then formulate the optimization problem for the EQ_DELAY policy as follows:

Problem 2 (EQ_DELAY). Find the server access probability vector

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p} \in (\mathcal{P} \cap \mathcal{S})} \left(\overline{T}(\mathbf{p})\right).$$

As with Problem 1, it is fairly easy to verify that the set of inactive servers $I(\hat{\mathbf{p}})$ must be one of the following: $\emptyset, \{N\}, \{N-1, N\}, \ldots, \{2, 3, \ldots, N\}$. Denote the slowest active server in the minimal solution of Problem 2 as $\hat{N}$. The following theorem, which is the analog of Theorem 2, provides expressions for $\hat{N}$ and $\hat{\mathbf{p}}$:

Theorem 3. The solution $\hat{\mathbf{p}}$ to Problem 2 can be obtained as follows. Define

$$\beta_i = \frac{\lambda + i\mu_i - \sum_{j=1}^{i} \mu_j}{i\lambda}, \quad 0 \le i \le N. \tag{7}$$

Then,

1. $\hat{N}$ is the maximum index i for which $\beta_i > 0$.
2. $I(\hat{\mathbf{p}}) = \{\hat{N}+1, \hat{N}+2, \ldots, N\}$,
3. $\hat{p}_i = \frac{\lambda + \hat{N}\mu_i - \sum_{j=1}^{\hat{N}} \mu_j}{\hat{N}\lambda} \quad \forall i \notin I(\hat{\mathbf{p}})$,
4. $\hat{p}_i = 0 \quad \forall i \in I(\hat{\mathbf{p}})$.

We note that the algorithm for the OPT policy in Section 3 can easily be modified to compute $\hat{N}$ and $\hat{\mathbf{p}}$.

The average delay of requests for the EQ_DELAY policy is obtained by inserting the derived expression of $\hat{\mathbf{p}}$ into Eq. (2)

$$\overline{T}(\hat{\mathbf{p}}) = \sum_{i=1}^{\hat{N}} \frac{\hat{p}_i}{\mu_i - \hat{p}_i\lambda} = \frac{\hat{N}}{\sum_{i=1}^{\hat{N}} \mu_i - \lambda}. \tag{8}$$

4.2 General Solution for EQ_LOAD

The EQ_LOAD policy aims at achieving the same utilization at each of the servers in the system. The problem can be formally stated as follows:

Problem 3 (EQ_LOAD). Find a server access probability vector $\tilde{\mathbf{p}} \in \mathcal{P}$ such that

$$\frac{\tilde{p}_i}{\mu_i} = \frac{\tilde{p}_j}{\mu_j} \quad \forall i, j = 1, 2, \ldots, N.$$

The solution to this problem is straightforward and is given by

$$\tilde{p}_i = \frac{\mu_i}{\sum_{j=1}^{N} \mu_j} \quad \forall i = 1, 2, \ldots, N. \tag{9}$$

The average delay for EQ_LOAD satisfies the following expression

$$\overline{T}(\tilde{\mathbf{p}}) = \frac{N}{\sum_{i=1}^{N} \mu_i - \lambda}. \tag{10}$$

4.3 Comparison

In this section, we compare the performance of the OPT, EQ_DELAY and EQ_LOAD policies. Our first observation is that EQ_DELAY and EQ_LOAD have the same average delay when $\hat{N} = N$, because Eq. (8) becomes identical to Eq. (10) in this case. Therefore, EQ_DELAY and EQ_LOAD always perform comparably at high load, since EQ_DELAY must use all the servers to ensure stability in this regime. This result is summarized by the following lemma.

Lemma 1. *If $\hat{N} = N$, then $\overline{T}(\hat{\mathbf{p}}) = \overline{T}(\tilde{\mathbf{p}})$.*

We next show, that for a given number of servers N, there exist arrival rate and service rate parameters such that the ratio of the average delay of EQ_DELAY to that of OPT approaches N. This situation occurs at high load, when $\hat{N} = N$. Therefore, from Lemma 1 the same result applies for the ratio of the average delay of EQ_LOAD to that of OPT.

Theorem 4. For any given N, there exist parameters λ and μ_i, $i = 1, 2, \ldots, N$, such that $\overline{T}(\tilde{\mathbf{p}})/\overline{T}(\mathbf{p}^*) = \overline{T}(\hat{\mathbf{p}})/\overline{T}(\mathbf{p}^*) \to N$.

Proof. For any N, we derive an example with a particular configuration of arrival rate and service rates parameters such that the average delay ratio tends to N. Assume the sum of service rates of all the servers is 1. Furthermore, assume μ_1 is close to 1 and $\mu_1 \gg \mu_2 = \mu_3 = \cdots = \mu_N$. Thus, $\mu_i = \frac{1-\mu_1}{N-1}$ for $1 < i \leq N$. In addition, assume $\sum_{i=1}^{N-1} \mu_i < \lambda < 1$. Then $N^* = \hat{N} = N$ and $\overline{T}(\tilde{\mathbf{p}}) = \overline{T}(\hat{\mathbf{p}})$. The ratio of Eq. (8) to Eq. (5) yields

$$\begin{aligned}\frac{\overline{T}(\hat{\mathbf{p}})}{\overline{T}(\mathbf{p}^*)} &= \frac{N\lambda}{\left(\sum_{i=1}^{N^*} \sqrt{\mu_i}\right)^2 - N\left(\sum_{i=1}^{N} \mu_i - \lambda\right)} \geq \frac{N\lambda}{\left(\sum_{i=1}^{N} \sqrt{\mu_i}\right)^2} \\ &= \frac{N\lambda}{\left(\sqrt{\mu_1} + \sqrt{(N-1)(1-\mu_1)}\right)^2}.\end{aligned} \tag{11}$$

Note that for any given N, the expression $(N-1)(1-\mu_1)$ appearing in the denominator of Eq. (11) becomes arbitrarily small as $\mu_1 \to 1$. Since $\mu_1 < \lambda < 1$, we also have that $\lambda \to 1$ as $\mu_1 \to 1$. Therefore, Eq. (11) becomes arbitrarily close to N as $\mu_1 \to 1$ and the theorem is proven. □

5 Simulations

In this section, we perform extensive simulations to validate and compare the OPT policy versus EQ_DELAY and EQ_LOAD. We first consider the case where the workload conforms to our $M/G/1$-PS model. We assume that the file size (and hence the service time) follows a heavy-tailed Pareto distribution with cumulative distribution function

$$F(x) = 1 - \frac{1}{(1+ax)^b} \quad x \geq 0,$$

where b is the *Pareto tail index*. This choice is justified by the large number of experimental studies which have shown that Web file size distributions follow a Pareto distribution [20, 21]. Using this workload model, we compare the average delay and standard deviation of the delay (which relates to fairness) of the three policies.

Next, we evaluate the performance of the three policies using synthetic traces generated by a Web workload generator, called ProWGen [10]. These traces exhibit temporal locality in file popularity and, therefore, differ from the $M/G/1$-PS model. Our simulations shows that OPT outperforms substantially the two other policies in this case as well.

5.1 Average Delay

We first compare the average delay of requests with OPT, EQ_DELAY and EQ_LOAD as a function of the aggregate load ρ, in the system which is defined as $\rho = \lambda / \sum_{j=1}^{N} \mu_j$. We fix the sum of the server rates to be 100 KB/s. We assume that the system consists of $N = 5$ servers, with one of them eight times faster than the others. Thus, the fastest server has service rate 66.67 KB/s, while the others have rate 8.33 KB/s. For the file size distribution, we assume a Pareto distribution with tail index $b = 2.2$. The mean file size is 11 KB. Each simulation point represents the average over 50 runs. A simulation run consists of 0.5×10^6 requests.

Figure 1 depicts the analytical results and simulation results for the average delay using the OPT, EQ_DELAY, and EQ_LOAD policies. The analytical expressions are obtained respectively from Eqs. (5), (8), and (10). As expected the simulation results match the analytical expressions. The figure also displays 95% confidence intervals which turn out to be very small (for this reason, we do not show them in other figures).

From Fig. 1, we observe that both OPT and EQ_DELAY perform much better than EQ_LOAD at low load. The reason is that, in this regime, OPT and EQ_DELAY forward requests only to the fastest server, while EQ_LOAD always sends $1/3$ of the requests to the slower servers. At high load, EQ_LOAD and EQ_DELAY achieve the same average delay, as predicted by Lemma 1, and OPT performs significantly better than these two policies. For instance, at load $\rho = 0.9$, the average delay of EQ_DELAY and EQ_LOAD is approximately 30% higher than the optimal average delay.

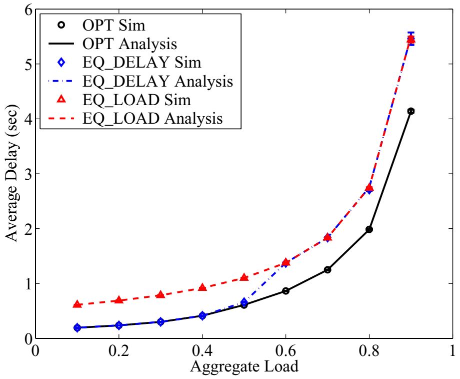


Fig. 1. Average delay: simulation and analysis

5.2 Standard Deviation of Delay and Fairness

An apparent advantage of EQ_DELAY is to serve each request with the same average delay (at the cost of higher overall average delay). This might lead to the belief that it exhibits better fairness properties than the other server selection policies. However, we show that this is actually not the case, at least with respect to OPT.

In the following set of simulations, we evaluate the standard deviation of the delay obtained with each policy. This metric has been recognized as one of the best ways to measure fairness in queues [22]. We consider a system consisting of $N = 10$ servers with service rate vector (188.2 65.18 40.71 25.13 20.72 19.53 15.87 14.63 8.67 6.41) KB/s. The Pareto tail index is $b = 2.2$ and the mean file size is 40 KB.

Figure 2 depicts the results. We observe that OPT and EQ_DELAY perform very similarly, while the standard deviation of EQ_LOAD is noticeably higher at all utilization values. A few comments are in order here. First, even though the average delay at each server is the same for EQ_DELAY, the actual delay of a request is still a random variable. In particular, this delay depends on the number of other requests concurrently being served, which obviously varies over time. Therefore, the standard deviation of EQ_DELAY is non-zero and turns out to be on the same order as that of OPT. Second, the standard deviation of EQ_LOAD is higher than that of EQ_DELAY, even at high load. This result is somewhat surprising since the average delay of these two policies is identical in this regime. This discrepancy is resolved by noting that the average delay of requests at different servers is not the same for EQ_LOAD. Therefore, EQ_LOAD and EQ_DELAY are not statistically identical even at high load. Finally, we observe that the standard deviation is quite high with all the policies. This is due to the extremely high variability of the file sizes generated by the Pareto distribution.

5.3 ProWGen Simulations

Our analytic $M/G/1$ Processor Sharing model and solutions are based on the assumption that the service time of different requests is independent. On the other hand, some experimental studies have shown that Web requests exhibits temporal locality, see

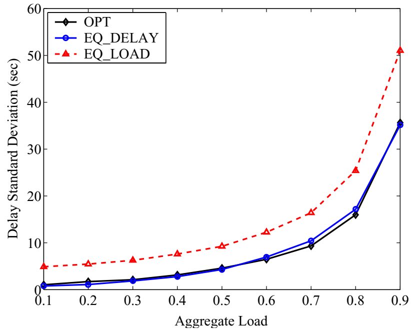


Fig. 2. Standard deviation of delay

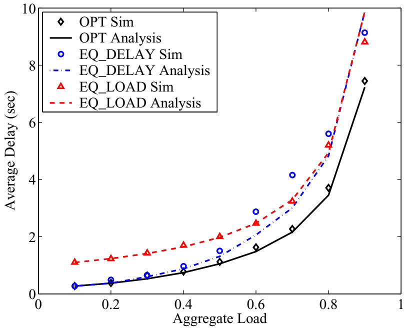


Fig. 3. Average delay with ProWGen traces

e.g. [23]. In this section, we examine how OPT, EQ_DELAY and EQ_LOAD perform when request streams have short-term temporal correlation.

We use the Web workload generator ProWGen [10] to produce synthetic Web workload that exhibits temporal locality in file popularity. ProWGen, originally developed for Web cache performance evaluation, models the file popularity using Zipf distribution, and file size distribution using a combination of lognormal body and a Pareto tail. It can also model positive or negative correlation between file size and popularity if needed. Furthermore, it can use an LRU stack to model request temporal locality.

In our simulations, we use default values for most ProWGen parameters, with the Zipf slope of 0.75, the Pareto Tail index of 1.2, the lognormal mean of 7KB and the standard deviation of 11KB, and the tail cutoff size of 10KB. We also have a "dynamic" stack with a depth of 1000 requests to introduce temporal locality. Finally, the correlation between file size and popularity is set to zero, which is consistent with literature findings [24].

We find that a peculiarity of ProWGen is that popular files tend to be generated at the end of the trace, and "one-timers" (files accessed only once) are more likely to be generated earlier in the trace. To mitigate this artifact while still maintaining most of the short-term temporal locality, we divide the trace generated by ProWGen into segments each consisting of 4,000 requests and reshuffle the segments to obtain the trace used in our simulations. We scale the mean file size to 40 KB, and use the same service rate array as in Section 5.2. Each run contains about 550,000 requests, and the results are averaged over 50 runs.

The simulation results are shown in Figure 3 together with analytical values based on our $M/G/1$-PS model.

We make the following observations from the results obtained in these experiments. First, OPT still achieves the minimal average delay, which is substantially smaller than that of EQ_LOAD or EQ_DELAY at high load. Second, simulation results match the analysis fairly well for OPT and EQ_LOAD, but poorly for EQ_DELAY. This is probably because of the temporal locality of ProWGen-generated traces. Even after the reshuffling, the moving average file size can significantly deviate from the long-term average. Since EQ_DELAY is very sensitive to load estimation errors [19], its performance suffered the most in this dynamic environment.

6 Conclusions

In this work, we have investigated the problem of optimal server selection in content replication networks. This problem has gained significant importance in recent years with the deployment of large-scale content delivery and peer-to-peer network architectures over the Internet. For this purpose, we have introduced a mathematical framework, based on the $M/G/1$ Processor Sharing queueing model, which allowed us to provide quantitative, yet non-trivial, insight into the performance server selection policies. We have also provided justification to our modeling assumptions based on measurement studies reported in the literature.

Based on our modeling assumptions, we have derived closed-form expressions for the optimal server access probabilities and the optimal average delay. We have also proposed a simple algorithm of linear computational complexity $O(N)$ to compute these probabilities, where N denotes the total number of servers in the system.

We have also evaluated the performance of two widely deployed server selection policies, generically referred to as EQ_DELAY and EQ_LOAD, that are representative of a large class of existing algorithms. In particular, we have analytically proved that the average delay of EQ_DELAY or EQ_LOAD can be as much as N times larger than the optimal delay. Thus, an important theoretical contribution of this work has been to show that the performance difference between EQ_DELAY (or EQ_LOAD) and OPT is unbounded as N grows. This radical difference in performance can be observed at high load and when the service capacity across servers is highly heterogeneous. Another interesting finding from our analysis is that EQ_DELAY and EQ_LOAD have identical average delay at high load, although their delay variance is different.

We have also conducted extensive simulations that demonstrated the significant superiority of OPT over EQ_LOAD and EQ_DELAY, especially at high load. In particular, we have shown that OPT has the potential of reducing the average delay in this regime by as much as 30%. We have also evaluated the degree of fairness of the three policies, that was quantified by computing the standard deviation of the delay, and shown that OPT performs comparably to EQ_DELAY and much better than EQ_LOAD.

The actual implementation of the OPT policy into real network settings is an important area of research left for future work. However, our initial results in this paper are extremely promising with this regard. In particular, we have shown that the performance of OPT is not too sensitive to the workload model, as illustrated by our simulations results obtained with the ProWGen workload generator.

References

1. Obraczka, K., Danzig, P., DeLucia, D.: Massively replicating services in autonomously managed, wide-area internetworks (1993) University of Southern California - Technical Report No. 93-541.
2. Vakali, A., Pallis, G.: Content delivery networks: Status and trends. IEEE Internet Computing **7** (2003) 68–74
3. Yang, B., Molina, H.G.: Designing a super-peer network. In: Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India (2003)

4. Cardellini, V., Casalicchio, E., Colajanni, M., Yu, P.: The state of the art in locally distributed web-server systems. ACM Computing Surveys (CSUR) **34** (2002) 263–311
5. Zegura, E., Ammar, M., Fei, Z., Bhattacharjee, S.: Application-layer anycasting: a server selection architecture and use in a replicated web service. IEEE/ACM Transactions on Networking **8** (2000)
6. Cisco: The global server load balancing primer. (http://www.cisco.com/en/US/products/hw/contnetw/ps4162/products_white_paper09186a00801b7725.shtml)
7. Korilis, Y., Lazar, A., Orda, A.: Architecting noncooperative networks. IEEE Journal of Selected Areas in Communications **13** (1995) 1241–1251
8. Stemm, M., Katz, R., Seshan, S.: A network measurement architecture for adaptive applications. In: Proceedings of IEEE INFOCOM, Tel-Aviv,Israel (2000)
9. Villela, D., Pradhan, P., Rubenstein, D.: Provisioning servers in the application tier for e-commerce systems. In: Proceedings of IWQoS '04. (2004)
10. Busari, M., Williamson, C.: ProWGen: a synthetic workload generation tool for simulation evaluation of Web proxy caches. Computer Networks **38** (2002)
11. Kleinrock, L.: Queueing systems. Volume 2. Wiley (1976)
12. Apache: (http://www.apache.org/)
13. Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, C.: Packet-level traffic measurements from the Sprint IP backbone. IEEE Network Magazine **17** (2003) 6–16
14. Ranjan, S., Karrer, R., Knightly, E.: Wide area redirection of dynamic content by Internet data centers. In: Proceedings of IEEE INFOCOM, Hong Kong, China (2004)
15. Tantawi, A., Towsley, D.: Optimal static load balancing in distributed computer systems. Journal of the ACM **32** (1985) 445–465
16. Kim, C., Kameda, H.: An algorithm for optimal static load balancing in distributed computer systems. IEEE Transactions on Computers **41** (1992) 381–384
17. Gerla, M., Kleinrock, L.: On the topological design of distributed computer networks. IEEE Transactions on Communications **25** (1977) 48–60
18. Bertsekas, D.: Nonlinear Programming. second edn. Athena Scientific (1999)
19. Starobinski, D., Wu, T.: Server selection for scalable internet services: Algorithms and analysis. http://www.bu.edu/systems/research/publications/2004/2004-IR-0020.pdf (2004) Boston University - Technical Report No. 2004-IR-0020.
20. Crovella, M., Bestavros, A.: Self-similarity in world wide web traffic: Evidence and possible causes. IEEE/ACM Trans. on Networking **5** (1997) 835–846
21. Starobinski, D., Sidi, M.: Modeling and analysis of power-tail distributions via classical teletraffic methods. Queueing Systems (QUESTA) **36** (2000) 243–267
22. Avi-Itzhak, B., Levy, H.: On measuring fairness in queues. Advances of Applied Probability (2004)
23. Mahanti, A., Eager, D., Williamson, C.: Temporal locality and its impact on Web proxy cache performance. Performance Evaluation **42** (2000) 187–203
24. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and zipf-life distributions: evidence and implications. In: Proceedings of IEEE INFOCOM '99, New York, NY (1999) 126–134

Local Utility Aware Content Replication*

Nikolaos Laoutaris, Orestis Telelis, Vassilios Zissimopoulos,
and Ioannis Stavrakakis

Department of Informatics and Telecommunications,
University of Athens, 15784 Athens, Greece
{laoutaris, telelis, vassilis, ioannis}@di.uoa.gr

Abstract. A commonly employed abstraction for studying the object placement problem for the purpose of Internet content distribution is that of a distributed replication group. In this work the initial model of distributed replication group of Leff, Wolf, and Yu (IEEE TPDS '93) is extended to the case that individual nodes act selfishly, i.e., cater to the optimization of their individual local utilities. Our main contribution is the derivation of equilibrium object placement strategies that: (a) can guarantee improved local utilities for all nodes concurrently as compared to the corresponding local utilities under greedy local object placement; (b) do not suffer from potential mistreatment problems, inherent to centralized strategies that aim at optimizing the social utility; (c) do not require the existence of complete information at all nodes. We develop a baseline computationally efficient algorithm for obtaining the aforementioned equilibrium strategies and then extend it to improve its performance with respect to fairness. Both algorithms are realizable in practice through a distributed protocol that requires only limited exchange of information.

1 Introduction

A commonly employed abstraction for studying content distribution systems is that of a distributed replication group [1]. Under this abstraction, nodes utilize their storage capacity to replicate information objects in order to make them available to local and remote users. A request issued by a local user and serviced locally (i.e., involving a locally replicated object), is served immediately, thus incurring a minimal cost. Otherwise, the requested object is searched in other nodes of the group and if not found, it is retrieved from the origin server; the access cost, however, increases with the distance. Depending on the particular application, the search for objects at remote nodes may be conducted through query protocols, succinct summaries, DNS redirection or distributed hash tables.

Several placements problems can be defined regarding a distributed replication group. The proxy (or cache, or mirror, or surrogate) placement problem

* This work and its dissemination efforts have been supported in part by the IST Programs of the European Union under contracts IST-6475 (ACCA) and FP6-506869 (E-NEXT).

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 455–468, 2005.

refers to the selection of appropriate physical network locations (routers) for installing content proxies [2, 3]. Another relevant problem is the object placement problem, which refers to the selection of objects for the nodes, under given node locations and capacities [1, 4, 5]. Joint formulations of the above mentioned problems have also appeared, e.g. in [6, 7], where the proxy placement, proxy dimensioning, and object placement problems are combined into a single problem.

All the aforementioned work has focused on the optimization of the so called *social utility* (sum of the individual *local utilities* of the nodes, defined as the delay and bandwidth gains from employing replication). Optimizing the social utility is naturally the objective in environments where a central authority dictates its replication decisions to the nodes. It suits well applications such as web mirroring and CDNs, which are operated centrally by a single authority (this being the content creator or the content distributor). Applications that are run by multiple authorities, such as web caching networks and P2P networks, may also seek to optimize the social utility. This, however, requires some nodes to act in a spirit of voluntaryism, as the optimization of the social utility is often harmful to several local utilities.

Consider as an example a group of nodes that collectively replicate content. If one of the nodes generates the majority of the requests, then a socially optimal (SO) object placement strategy will use the storage capacity of other nodes to replicate objects that do not fit in the over-active node's cache. Consequently, the users of these other nodes will experience a service deterioration as a result of their storage being hijacked by potentially irrelevant objects with regard to their local demand. In fact, such nodes would be better served if they acted independently and employed a greedy local (GL) object placement strategy (i.e., replicated the most popular objects according to the local demand). A similar situation can arise if caching, rather that replication, is in place: remote hits originating from other nodes may evict objects of local interest in an LRU-operated cache that participates in a web caching network. Concern for such exploitation can prevent rational nodes from participating in such groups, and, instead, lead them to operating in isolation in a greedy local manner. Such a behavior, however, is far from being desirable.

Being GL is often ineffective in terms of performance, not only with respect to the social utility, but with respect to the individual local utilities too. For example, when the nodes have similar demand patterns and the inter-node distances are small, then replicating multiple times the same most popular objects, as done by the same repeated GL placement at all the nodes, is highly ineffective. Clearly, all the nodes may gain substantially in this case, if they cooperate and replicate different objects. In fact, it is even possible that an appropriate cooperation of the nodes can lead to a simultaneous improvement of all local utilities as compared to the GL performance. However, nodes cannot recognize such opportunities for mutually beneficial cooperation, since they are generally unaware of the remote demand patterns. On the other hand, they cannot know the impact (bad or good) that the SO object placement strategy may have on their own local utility.

To address the above mentioned deadlock, we use as reference the object replication problem defined by Leff et al. [1], and extend it to account for the existence of selfishly motivated nodes. We use a strategic game in normal form [8] to model the contention between the selfish nodes and set out to identify pure Nash equilibrium object placement strategies (henceforth abbreviated EQ). There are several advantages in employing EQ strategies. First, by their definition, they can guarantee for each and every node of the group that its local utility under EQ will be at least as good as under GL, and possibly better. The first case ("at least as good") precludes mistreatment problems such as those that can arise under the SO placement which cause the nodes to leave the group in pursuit of GL placement. The second case ("possibly better") is the typical one, and points to the fact that implicit cooperation is induced even by selfishly behaving nodes as they attempt to do better than GL. Consequently, the EQ strategy is in position to break the above mentioned deadlock, as it forbids the mistreatment of any one node, while it also guards against the disintegration of the group, and the poor performance associated with the GL strategy.

Our main result is that such EQ object placement strategies can be obtained by simple distributed algorithms that do not require the existence of complete information at all the nodes. We describe a two-step local search (TSLS) algorithm for this purpose. TSLS requires each node to know only its local demand pattern and the objects selected for replication by remote nodes, but not the remote demand patterns of other nodes (the demand pattern of a node defines explicitly its utility function, thus in the presented framework it is not assumed that nodes know the utility functions of other nodes). Knowing the remote demand patterns requires the transmission of too much information and thus is seldom possible in large distributed replication groups. On the other hand, knowing the objects selected for replication by remote nodes requires the exchange of much less information, which can be reduced further by employing simple encoding schemes such as Bloom filters [9] (see also [10] for real distributed applications/protocols that utilize such information). Thus in terms of the required information, the proposed EQ strategies fit between the GL strategy that requires only local information, and the SO strategy that requires complete information.

A recent work on game theoretic aspects of caching and replication that is relevant to our work is due to Chun et al. [11]. However, this work does not consider storage capacity limits on the nodes and, thus, differs substantially from our approach.

The remainder of the article is structured as follows. Section 2 describes formally the distributed replication group and the distributed selfish replication (DSR) game. Section 3 describes the baseline TSLS object placement algorithm. Section 4 establishes that the TSLS algorithm produces a pure Nash equilibrium object placement strategy for the DSR game. Section 5 is devoted to the presentation of the TSLS(k) algorithm, whose development is largely motivated by our desire to obtain EQ placements without having to resort to the logical ordering of the nodes. Section 6 describes a distributed protocol for implementing the two algorithms. Section 7 presents some numerical examples. Finally, Section 8 concludes the article. The omitted proofs for the presented Propositions, as well as implementation details and more numerical results, can be found in a longer version of this article [12].

2 Definitions

Let o_i, $1 \leq i \leq N$, and v_j, $1 \leq j \leq n$, denote the ith unit-sized object and jth node, and let $O = \{o_1, \ldots, o_N\}$ and $V = \{v_1, \ldots, v_n\}$ denote the corresponding sets. Node v_j is assumed to have a storage capacity for C_j unit-sized objects and a demand described by a rate vector r_j over O, $r_j = \{r_{1j}, \ldots, r_{Nj}\}$, where r_{ij} denotes the rate (requests per second) at which node v_j requests object o_i; let also $\rho_j = \sum_{o_i \in O} r_{ij}$ denote the total request rate from v_j. We follow the access cost model defined in [1] and according to which, accessing an object from a node's local cache costs t_l, from a remote node's cache t_r, and from the origin server t_s, with $t_l \leq t_r \leq t_s$ (Fig. 1 depicts the envisaged distributed replication group).

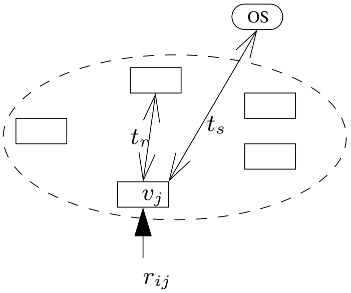


Fig. 1. A distributed replication group

Let $R_j = \{o_i \in O : r_{ij} > 0\}$ denote the request set of node v_j. Let P_j denote the placement of node v_j, that is the set of objects replicated at this node; $P_j \subseteq O$ and $|P_j| = C_j$. Let $P = \{P_1, P_2, \ldots, P_n\}$ be referred to as a global placement and let $P_{-j} = P_1 \cup \ldots \cup P_{j-1} \cup P_{j+1} \cup \ldots \cup P_n$ denote the set of objects collectively held by nodes other than v_j under the global placement P. The gain for node v_j under P is defined as follows:

$$G_j(P) = \sum_{o_i \in P_j} r_{ij} \cdot (t_s - t_l) + \sum_{\substack{o_i \notin P_j \\ o_i \in P_{-j}}} r_{ij} \cdot (t_s - t_r) \tag{1}$$

Such a gain definition captures the distance savings when accessing objects from nodes in the group (either the local node or remote ones) instead of the origin server, assumed to be the furthest away.

In the sequel, we define a game that captures the dynamics of distributed object replication under selfishly behaving nodes.

Definition 1. (DSR game) The distributed selfish replication game is defined by the tuple $\langle V, \{P_j\}, \{G_j\}\rangle$, where:

- V is the set of n players, which in this case are the nodes.
- $\{P_j\}$ is the set of strategies available to player v_j. As the strategies correspond to placements, player v_j has $\binom{N}{C_j}$ possible strategies.
- $\{G_j\}$ is the set of utilities for the individual players. The utility of player v_j under the outcome P, which in this case is a global placement, is $G_j(P)$.

DSR is a n-player, non-cooperative, non-zerosum game [8]. For this game, we seek equilibrium strategies, and in particular, pure Nash equilibrium strategies.

Definition 2. (pure Nash equilibrium for DSR) A pure Nash equilibrium for DSR is a global placement P^*, such that for every node $v_j \in V$:

$$G_j(P^*) \geq G_j((P_1^*, \ldots, P_{j-1}^*, P_j, P_{j+1}^*, \ldots, P_n^*)) \quad \text{for all } P_j \in \{P_j\}$$

That is, under such a placement P^*, nodes cannot modify their individual placements unilaterally and benefit. In the sequel, we develop polynomial time algorithms that, given an instance of the DSR game, can produce several Nash equilibrium placement strategies for it.

3 A Two-Step Local Search Algorithm

In this section we present a two-step local search algorithm that computes a placement for each one of the nodes. In Section 4 we show that these placements correspond to a Nash equilibrium global placement, that is they are EQ strategies. In Section 5, we modify the two-step local search algorithm in order to overcome some of its limitations.

Let P_j^0 and P_j^1 denote the GL placement strategy and the placement strategy identified by TSLS for node v_j, respectively. Let also $Greedy_j(\mathcal{P})$ denote a function that computes the optimal placement for node v_j, given the set $\mathcal{P}$ of distinct objects collectively held by other nodes; we elaborate on this function later on in the section. Table 1 outlines the proposed TSLS algorithm.

At the initialization step (Step 0) nodes compute their GL placements P_j^0. This is done by evaluating $Greedy_j(\emptyset)$ for each v_j, capturing the case in which nodes operate in isolation ($\mathcal{P} = \emptyset$).

Table 1. The TSLS algorithm

Step 0 (initialization): $P_j^0 = Greedy_j(\emptyset)$, $1 \leq j \leq n$.
Step 1 (improvement): $P_j^1 = Greedy_j(P_{-j}^{1^-})$, $1 \leq j \leq n$,
where, $P_{-j}^{1^-} = P_1^1 \cup \ldots \cup P_{j-1}^1 \cup P_{j+1}^0 \cup \ldots \cup P_n^0$

At the improvement step (Step 1) nodes observe the placements of other nodes and, based on this information, proceed to improve their own. The order in which nodes take turn in improving their initial placements is determined based on their ids (increasing order). Thus at v_j's turn to improve its initial placement, nodes $v_1, \ldots, v_{j-1}$ have already improved their own, while nodes $v_{j+1}, \ldots, v_n$, have not as yet done so. Node v_j obtains its improved placement P_j^1 by evaluating $Greedy_j(P_{-j}^{1^-})$, where $P_{-j}^{1^-} = P_1^1 \cup \ldots \cup P_{j-1}^1 \cup P_{j+1}^0 \cup \ldots \cup P_n^0$ denotes the set of distinct objects collectively held by other nodes (hence the $-j$ subscript) at the time prior to v_j's turn at Step 1 (hence the 1^- superscript).

We return now to describe how to compute the optimal placement for node v_j, when the set of distinct objects collectively held by other nodes is $\mathcal{P}$; such an optimization is employed twice by the TSLS algorithm: at Step 0 where $\mathcal{P} = \emptyset$, and at Step 1 where $\mathcal{P} = P_{-j}^{1^-}$. To carry it out, one has to select objects according to their relative excess gain, up to the limit set by the storage capacity of the node. Let g_{ij}^k denote the excess gain incurred by node v_j from replicating object o_i at step $k \in \{0, 1\}$ of TSLS; g_{ij}^k depends on v_j's demand for o_i and also on whether o_i is replicated elsewhere in the group.

$$g_{ij}^k = \begin{cases} r_{ij} \cdot (t_s - t_l) \text{ ,if} & k = 0 \quad \text{or} \quad k = 1, o_i \notin P_j^0, o_i \notin P_{-j}^{1^-} \\ r_{ij} \cdot (t_r - t_l) \text{ ,if} & k = 1, o_i \notin P_j^0, o_i \in P_{-j}^{1^-} \\ 0 \qquad\qquad\quad \text{ ,if} & k = 1, o_i \in P_j^0 \end{cases} \tag{2}$$

$r_{ij} \cdot (t_s - t_l)$ is the excess gain for v_j from choosing to replicate object o_i that is currently not replicated at any node in V. If o_i is replicated at some other node(s), then v_j's excess gain of replicating it locally is lower, and equal to $r_{ij} \cdot (t_r - t_l)$. Finally, there is no excess gain from choosing to replicate an object that is already replicated locally. Such excess gains are determined by the request frequency for an object, multiplied by the reduction in access cost achieved by fetching the object locally instead from the closest node that currently replicates it (either some other node in V or the origin server).

Finding the optimal placement for v_j given the objects replicated at other nodes ($\mathcal{P}$) amounts to solving a special case of the 0/1 Knapsack problem, in which object values are given by Eq. (2), object weights are unit, and the Knapsack capacity is equal to an integer value C_j. The optimal solution to this problem is obtained by the function $Greedy_j(\mathcal{P})$. This function first orders the N objects in a decreasing order according to g_{ij}^k ($k = 0$ at Step 0 and 1 at Step 1), and then it selects for replication at v_j the C_j most valuable ones.[1] As the objects are of unit size and the capacity is integral, this greedy solution is guaranteed to be an optimal solution to the aforementioned 0/1 Knapsack problem.

[1] Ties are solved arbitrarily at Step 0 and not re-examined at Step 1, i.e., an object that yields the same gain as other objects and is selected at Step 0, is not replaced in favor of any one of these equally valuable objects, later on at Step 1. Thus at Step 1, new objects are selected only if they yield a gain that is strictly higher than that of already selected ones.

We now proceed to connect the 0/1 Knapsack problem under the g_{ij}^k's, with the gain $G_j(\cdot)$ for v_j under a global placement. We will show that solving the 0/1 Knapsack for v_j under given $P_{-j}^{1^-}$ is equivalent to maximizing $G_j(\cdot)$, given the current placements of nodes other than v_j.

Proposition 1. The placement $P_j^1 = Greedy_j(P_{-j}^{1^-})$ produced by the TSLS algorithm for node v_j, $1 \leq j \leq n$, satisfies:

$$G_j(P_1^1,\ldots,P_{j-1}^1,P_j^1,P_{j+1}^0,\ldots,P_n^0) \geq G_j(P_1^1,\ldots,P_{j-1}^1,P_j,P_{j+1}^0,\ldots,P_n^0), \forall P_j \in \{P_j\}.$$

The proof for Proposition 1 and subsequent ones can be found in a longer version of this article [12].

An important observation is that at the improvement step, a node is allowed to retain its initial GL placement, if this is the placement that maximizes its gain given the placements of other nodes. Thus, the final gain of a node will be at least as high as its GL one, irrespectively of the demand characteristics of other nodes; this eliminates the possibility of mistreatment due to the existence of overactive nodes. Regarding the complexity of TSLS, we show in [12] that it is $O(nN \log N)$.

4 Existence of a Pure Nash Equilibrium for DSR

In this section it is shown that the global placement $(P_1^1, P_2^1, \ldots, P_n^1)$ produced by the TSLS algorithm is a pure Nash equilibrium of the distributed replication game. To prove this result we introduce the following additional definitions. Let $E_j^1 = \{o_i \in O : o_i \in P_j^0, o_i \notin P_j^1\}$ denote the eviction set of v_j at Step 1; it is a subset of the initial placement, comprising objects that are evicted in favor of new ones during v_j's turn to improve its initial placement. Similarly, $I_j^1 = \{o_i \in O : o_i \notin P_j^0, o_i \in P_j^1\}$ denotes the insertion set, i.e., the set of new objects that take the place of the objects that belong to E_j^1. At any point of TSLS an object is dubbed a *multiple*, if it is replicated in more than one nodes, and an *unrepresented one* (*represented one*), if there is no (some) node replicating it. Regarding these categories of objects, we can prove the following:

Proposition 2. (only multiples are evicted) The TSLS algorithm guarantees that the eviction set of node v_j is such that $E_j^1 \subseteq (P_j^0 \cap P_{-j}^{1^-})$.

Proposition 3. (only unrepresented ones are inserted) The TSLS algorithm guarantees that the insertion set of node v_j is such that $I_j^1 \subseteq (R_j/(P_j^0 \cup P_{-j}^{1^-}))$.

The previous two propositions enable us to prove that TSLS finds a pure Nash equilibrium for DSR.

Proposition 4. The global placement $P^1 = (P_1^1, P_2^1, \ldots, P_n^1)$ produced at the end of Step 1 of the TSLS algorithm is a pure Nash equilibrium for the distributed replication game.

Assuming that no two g_{ij}^{k} are the same, then the maximum number of different equilibria that may be identified by the TSLS algorithm is $n!$, i.e., a different equilibrium for each possible ordering (permutation) of the n nodes.

At this point we would like to discuss the subtle difference between the DSR game and the TSLS algorithm, which is just a solution for the DSR game, and not an augmented game that also models the ordering of nodes. The DSR game is a well defined game as it is, i.e., without reference to node ordering, or any other concept utilized by the particular TSLS solution. The ordering of nodes is hence just a device for deriving equilibrium placements, and not a concept of the DSR game itself. The ordering of nodes is not required for defining the DSR game.

The use of a specific ordering of nodes in the improvement step of TSLS is central to the algorithm's ability to find equilibrium placements. It might be the case that algorithms of complete information exist that can find equilibrium placements without requiring the use of such a device. For the case of a distributed replication group, however, it seems that some synchronization mechanism, like the ordering of nodes, is required in order to be able to implement a solution algorithm in a distributed manner without requiring complete information (remote utility functions). In [12] we show that by permitting the nodes to improve their placements without coordination with respect to order, they may never reach a stable placement, but instead loop indefinitely through various transient placements.

5 TSLS(k): Improving on the TSLS Fairness

Consider the case of a homogeneous group, i.e., a group composed of nodes with identical characteristics in terms of capacity, total request rate, and demand distribution. Since such nodes are identical, it is natural to expect that they will be treated equally by a placement strategy. Under the TSLS algorithm, however, the amount of gain that a node receives depends on the node's turn during the improvement step. When the different demand patterns are similar (which is the most interesting case because it allows for higher mutual benefit), then the nodes with the higher turns have an advantage as they can fully utilize the replication decisions of previous nodes [12][2]. This allows for the possibility that small differences in the "merit quantity" of nodes, based on which the turns are decided (to be defined later on in Sect. 6), will translate into large differences in the assigned node turns. This can lead to an unequal treatment of the individual homogeneous nodes.

To address this issue we propose the TSLS(k) algorithm, which is a variation of the baseline TSLS algorithm. Under TSLS(k), *each node may perform only up to k changes during a round of the improvement step*, i.e., evict up to k

[2] In the extended version we also show that a higher turn is not always better under arbitrary demand patterns. This means that the nodes cannot generally act strategically and determine their optimal turn.

objects to insert an equal number of new ones. Note that under TSLS, any number of changes are permitted during the single round of the improvement step. Under TSLS(k), the improvement step might require multiple rounds to reach an equilibrium placement, whereas under TSLS, an equilibrium placement is reached only after a single round. Intuitively, TSLS(k) works in a round-robin fashion based on some node ordering, and allows each node to perform up to k changes of its current placement during a given round, even if the node would like to perform more changes; for additional changes, the node has to wait for subsequent rounds. The effect of this round-robin, k-constrained selection of objects, is that TSLS(k) is at least as and generally more fair than TSLS with respect to the achieved individual gains. By selecting sufficiently small values of k, e.g., $k = 1$, which is an extreme case, *it is possible to almost eliminate the effect of a node's turn on the amount of gain that it receives under the final placement.* Essentially, when k is small, TSLS(k) is able to overcome the inherent limitations with respect to fairness that arise when having to decide a specific node ordering in order to produce an equilibrium placement. For k sufficiently large (approaching the maximum node capacity C^{max}), TSLS(k) reduces to the baseline TSLS. What the TSLS(k) algorithm provides is essentially a tradeoff between an increased fairness and an increased execution time due to the multiple rounds during the improvement step. In [12] we present the full implementation details of TSLS(k) as well as proofs for its convergence in a finite number of rounds upper bounded by $\lceil C^{max}/k \rceil$.

6 A Protocol for Applying the Nash Equilibrium for DSR

In this section we outline a protocol for implementing TSLS or TSLS(k) in a distributed replication group.

6.1 Deciding Turns for the Improvement Step

First, we describe a simple way for deciding turns that can be used with both TSLS and TSLS(k); for TSLS the ordering has an impact on the individual gains, whereas for TSLS(k) under small k, the ordering has a diminishing effect on the gains. Consider an arbitrary labelling of nodes, not related to the ordering in which nodes take turns. Let T_h denote a "merit" quantity associated with node v_h, $1 \leq h \leq n$, based on which v_h's turn is decided. T_h will be defined in such a way that larger ids (turns) will be assigned to nodes having larger values T_h. At the end, a node whose T_h value is the jth largest one, will be re-labelled v_j, thus taking the jth turn. There are many ways to define T_h for a node; among them the following three are of particular interest because they can be naturally associated with a common case in which nodes have similar demand patterns and where a higher turn is better (see Sect. 7):

$$T_h = \begin{cases} C_h & \text{(proportional fairness)} \\ \rho_h & \text{(pro social benefit)} \\ C_h \cdot \rho_h & \text{(hybrid)} \end{cases}$$

The first one caters to *proportional fairness*. It suggests that a node's turn, or equivalently its share of the extra gain produced through the cooperation, be proportional to the amount of resource (storage capacity) that the node contributes. Under such T_h, v_j is the jth largest node.

The second definition is a *socially* inclining one. It favors nodes that generate more requests, as these nodes have the largest influence on the social utility. Under such T_h, v_j is the jth more active node. Notice that following such a criterion for deciding turns is by no means equivalent to the SO strategy. An equilibrium placement under the pro social benefit criterion favors active nodes by allocating them a bigger share of the extra gain produced through the cooperation; this is to say that all other nodes will have (at least) their GL gain intact, whereas under SO, the benefited nodes may cause other nodes to fall below the GL level of gain.

The third expression for T_h is a hybrid way of splitting the gains of the cooperation; it favors nodes that contribute more storage and also produce more requests. Having defined the criterion based on which turns are decided, we move on to defining a protocol for implementing the algorithms and obtaining the equilibrium placement that corresponds to the decided ordering.

6.2 Distributed Protocol

A straightforward centralized implementation would require each node to report r_j and C_j to a central node responsible for executing the TSLS algorithm and sending back the placements P_j. The problem with such a centralized architecture is that it requires transmitting n rate vectors r_j, with each one containing N (object id, request probability) pairs; for large N this can lead to the consumption of too much bandwidth. We, therefore, turn our attention to the development of the following fully distributed protocol which involves three phases:

Phase DT: During this phase, turns are decided.

1. Each node v_h multicasts[3] to the group its value pair (C_h, ρ_h), while listening for, and storing, such pairs from other nodes. The truthfulness of the transmitted pair is crosschecked later on by other nodes during the operation of the distributed group.
2. Having listened to $n-1$ other value pairs, each node may compute its turn j based on a pre-agreed definition of T_h.

Phase 0: In this phase the initial placements according to TSLS are computed and distributed.

1. Each node v_j computes its initial placement P_j^0 and multicasts it to the group. Taking turns is not required at this phase and nodes may transmit their information concurrently.
2. Nodes listen and store the initial placements of other nodes.

[3] Native, or end-system, multicast can be employed.

Phase 1: In this phase, the initial placements of TSLS are improved.

1. Node v_j waits for its turn (i.e., until v_{j-1} completes its own turn and transmits) and then computes its improved placement P_j^1 as described by TSLS.
2. Following the computation of P_j^1, node v_j transmits E_j^1 and I_j^1 to the group.
3. Nodes $v_{j'}$, $j < j' \leq n$ receive E_j^1 and I_j^1 and use them to produce P_j^1 using also P_j^0, which they have from Phase 0.

To implement the TSLS(k) algorithm, Phase 1 needs to be repeated until no node has any more changes to perform. As was mentioned earlier, TSLS(k) provides a tradeoff between the improved fairness and the increased time required to perform multiple rounds at Phase 1. The volume of transmitted information, however, is essentially the same as with the baseline TSLS.

The aforementioned protocol has several advantages. It achieves a degree of parallelism, by permitting nodes to compute their initial placements during Phase 0 independently and concurrently with other nodes. Phase 1 involves a distributed computation too, albeit a sequential one. The major advantage, however, relates to the reduction in the amount of transmitted information as compared to a centralized computation which requires the transmission of $O(nN)$ pairs (object id, request frequency) towards the central point and then $O(\sum_{v_j \in V} C_j)$ object ids sent back from the central point to the nodes carrying the placements P_j^1. Our protocol limits[4] the amount of transmitted information to $O(\sum_{v_j \in V} C_j)$ object ids (initial placements plus eviction and insertion sets). This represents a substantial reduction in the amount of transmitted information, as typically the number of available objects is several orders of magnitude larger than the aggregate storage capacity of the group. Furthermore, lists of object ids can be represented succinctly by employing advanced compression techniques such as Bloom filters [9], whereas rate vectors composed of (object id, request frequency) elements, are much harder to represent and communicate.

With the above protocol, every node is aware of the placements of the other nodes. This information can be used for request routing as well as a means to detect and reveal untruthful nodes, i.e., nodes that try to exploit the group by declaring false placements (see [12] for details).

7 Numerical Examples

In this section we present a simple numerical example for the purpose of demonstrating the operation of TSLS and TSLS(k). Assume there exist two nodes that generate requests following the exact same Zipf-like distribution, i.e., $r_{ij} = \rho_j \cdot K/i^a$, where $K = (\sum_{i'=1}^{N} \frac{1}{i'^a})^{-1}$; the skewness parameter a captures the

[4] It is worthwhile to emphasize again that although the local placements are multicasted to the group, the amount of information that is gathered at each node is far less than the amount required by centralized algorithms (like the one for the SO placement). Here a node knows only the placements of other nodes, not the entire demand patterns of these nodes.

Table 2. An example with v_1, v_2 having the same Zipf-like demand pattern with $a = 0.8$. The number of available objects is $N = 100$ and the storage capacity of each node is $C = 40$. Also, $t_l = 0$, $t_r = 1$, $t_s = 2$, $\rho_1 = 1$

placement strategy	Node 1 objects	Node 2 objects
GL, $\rho_2 = X$	{1:40}	{1:40}
SO, $\rho_2 = 1$	$\{1:16\} \cup \{41:64\}$	{1:40}
SO, $\rho_2 = 2$	$\{1:12\} \cup \{41:68\}$	{1:40}
SO, $\rho_2 = 3$	$\{1:9\} \cup \{41:71\}$	{1:40}
SO, $\rho_2 = 4$	$\{1:7\} \cup \{41:73\}$	{1:40}
SO, $\rho_2 = 5$	$\{1:6\} \cup \{41:74\}$	{1:40}
SO, $\rho_2 = 6$	$\{1:5\} \cup \{41:75\}$	{1:40}
SO, $\rho_2 = 7$	$\{1:4\} \cup \{41:76\}$	{1:40}
SO, $\rho_2 = 8$	$\{1:4\} \cup \{41:76\}$	{1:40}
SO, $\rho_2 = 9$	$\{1:3\} \cup \{41:77\}$	{1:40}
SO, $\rho_2 = 10$	$\{1:3\} \cup \{41:77\}$	{1:40}
EQ, $\rho_2 = X$	$\{1:23\} \cup \{41:57\}$	{1:40}

degree of concentration of requests. The local access cost is, $t_l = 0$, the remote one, $t_r = 1$, and the cost of accessing the origin server, $t_s = 2$; this leads to a hop-count notion of distance. Finally, there are $N = 100$ distinct objects, and each node has a capacity for $C = 40$ objects.

In Table 2 we show the objects replicated under the GL, SO, and EQ replications strategies for fixed $\rho_1 = 1$ and varying ρ_2; here the EQ strategy is produced by the baseline TSLS. The GL strategy selects for each node the first 40 most popular objects, i.e., those with ids in {1:40}, independently of ρ_2. The SO strategy, however, is much different. As the request rate from Node 2 increases, SO uses some of the storage capacity of Node 1 for replicating objects that do not fit in Node 2's cache, thereby depriving Node 1 of valuable storage capacity for its own objects. For $\rho_2 = 10$, Node 1 gets to store only 3 of its most popular objects, while it uses the rest of its storage for picking up the next 37 more popular objects for Node 2, starting with the one with id 41. Under the EQ strategy Node 1 (v_1) stores 23 of its most popular objects. Node 2 (v_2) is the second one (i.e., the last one) to improve its placement, and it naturally selects the initial 40 most popular objects.

We now turn our attention to the gain G_j of the two nodes under the various placement strategies (the corresponding access cost can be obtained from the expression $t_s - G_j$). Figure 2 shows that as ρ_2 increases, the gain of v_2 under SO increases as it consumes storage from v_1 for replicating objects according to its preference; v_1's gain under SO decreases rapidly as a result of not being able to replicate locally some of its most popular objects. In fact, for $\rho_2 > 2.5$, v_1's gain becomes worse (lower) that the corresponding one under GL. From this point and onwards, v_1 is being mistreated by the SO strategy and thus has no incentive in participating in it, as it can obviously do better on its own under a GL placement.

By following an EQ strategy, a node's gain is immune to the relative request intensities and this is why the EQ lines are parallel to the x-axis of Fig. 2. v_1's

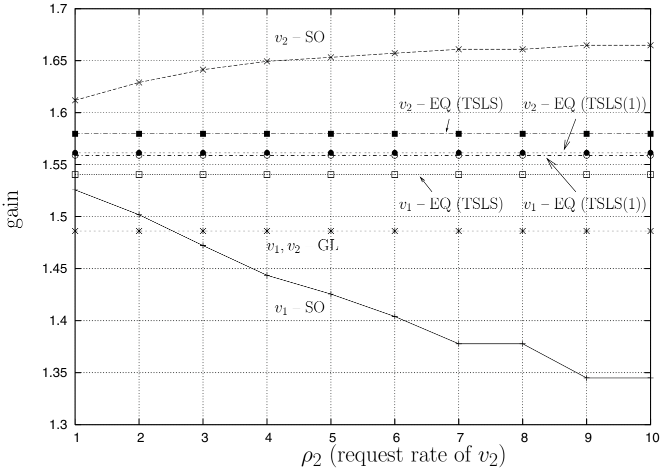


Fig. 2. Individual node gains for the example of Table 2. "v_j – XX" denotes the gain for node v_j under the placement strategy XX

gain under the EQ produced by TSLS is immune to the increasing ρ_2 and strictly higher than its gain under GL. This demonstrates the fact that the EQ strategy avoids the mistreatment problem. Under the EQ produced by TSLS both nodes achieve higher gains than with GL, but it is v_2 that benefits the most, and thus incurs a higher gain than v_1. This owes to the fact that v_2 is the second (last) one to improve its placement and, thus, has an advantage under TSLS. The difference in performance between the two nodes can be eliminated by employing the TSLS(k) algorithm. To show this, Fig. 2 includes the gains of the two nodes under the EQ strategy that is produced by TSLS(1). The corresponding lines almost coincide, which demonstrates the ability of TSLS(k) to be fair and to assign identical gains to v_1 and v_2 (as opposed to TSLS which, in this example, favors v_2).

8 Conclusions

This work has described two algorithms and an efficient distributed protocol for implementing equilibrium object placement strategies in a distributed selfish replication group. Such placement strategies become meaningful when replication nodes cater to their local utilities, as is the case with some content distribution applications that are run under multiple authorities (e.g., P2P, distributed web caching). In such applications, following a socially optimal placement strategy may lead to the mistreatment of some nodes, possibly causing their departure from the group. Our equilibrium strategies on the other hand, guarantee that all nodes are better off participating in the group as opposed to operating in isolation in a greedy local manner. This keeps a distributed group from splitting apart, by creating an excess gain for all (stemming from the cooperation) while forbidding the mistreatment of any one of the nodes. In a longer version of this article, we present several numerical examples for demonstrating the properties of the equilibrium placement strategies.

References

1. Avraham Leff, Joel L. Wolf, and Philip S. Yu, "Replication algorithms in a remote caching architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 11, pp. 1185–1204, Nov. 1993.
2. P. Krishnan, Danny Raz, and Yuval Shavit, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–581, Oct. 2000.
3. Bo Li, Mordecai J. Golin, Giuseppe F. Italiano, Xin Deng, and Kazem Sohraby, "On the optimal placement of web proxies in the internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.
4. Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman, "Placement algorithms for hierarchical cooperative caching," in *Proceedings of the 10th Annual Symposium on Discrete Algorithms (ACM-SIAM SODA)*, 1999, pp. 586 – 595.
5. Thanasis Loukopoulos and Ishfaq Ahmad, "Static and adaptive distributed data replication using genetic algorithms," *Journal of Parallel and Distributed Computing*, vol. 64, no. 11, pp. 1270–1285, Nov. 2004.
6. Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis, "Joint object placement and node dimensioning for internet content distribution," *Information Processing Letters*, vol. 89, no. 6, pp. 273–279, Mar. 2004.
7. Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis, "On the optimization of storage capacity allocation for content distribution," *Computer Networks*, vol. 47, no. 3, pp. 409–428, Feb. 2005.
8. Martin J. Osborne and Ariel Rubinstein, *A Course in Game Theory*, MIT Press, 1994.
9. Burton Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.
10. Andrei Broder and Michael Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Mathematics*, 2004, [accepted for publication].
11. Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz, "Selfish caching in distributed systems: A game-theoretic analysis," in *Proc. ACM Symposium on Principles of Distributed Computing (ACM PODC)*, Newfoundland, Canada, July 2004.
12. Nikolaos Laoutaris, Orestis Telelis, Vassilios Zissimopoulos, and Ioannis Stavrakakis, "Distributed selfish replication," UoA Technical Report, 2004, available on-line at: `http://www.cnl.di.uoa.gr/∼laoutaris/cgame_TECH.pdf`.

Improving Network Convergence Time and Network Stability of an OSPF-Routed IP Network

Amir Siddiqi[1] and Biswajit Nandy[2]

[1] Nortel, 3500 Carling Ave, Nepean, ON, K2H 8E9
amir@nortel.com
[2] Solana Networks
120 Robertson Road Suite 210, Nepean, ON, K2H 5Z1
bnandy@solananetworks.com

Abstract. The loss of routing protocol control messages due to network congestion can cause peer failures in routers, leading to routing failures and network instability. We attempt to decrease the network convergence time in response to a node or link failure in a network and study its implications on network stability. We develop an algorithm to dynamically tune OSPF failure threshold based on the network congestion level. The tunable OSPF protocol expedites failure detections as compared to standard OSPF protocol in lightly loaded networks by reducing the failure threshold parameter; whereas the failure threshold is increased during network congestion to sustain stable routing operation and avoid spurious failure decisions due to missed or delayed hello packets. Simulations performed with constant bit-rate (CBR) and FTP data traffic showed that the tunable OSPF protocol facilitated uninterrupted data transfer and resulted in shorter download times as compared to the standard OSPF protocol. Our findings demonstrate the importance of tunable failure threshold to achieve stable and robust network operation.

1 Introduction

The startling growth of the Internet and the flexibility of IP-based packet switched networks have expedited the convergence of data communications (packet switched) and voice-based communications (traditionally circuit switched) into a single IP-based core architecture. The Internet has become an interconnection of various types of heterogeneous networks and supports a wide range of applications. Aside from traditional web applications such as HTTP and FTP, the Internet is experiencing an exponential growth in audio and video streaming, and other real-time applications. However, the existing Internet only offers a single best-effort class of service that is mainly designed for traditional networks and applications, and can adversely impact the end-to-end performance of real-time applications and next-generation networks.

[1] This work is done as part of the Masters of Applied Sciences degree at the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada with the financial support of *The Natural Science and Engineering Research Council of Canada (NSERC).*

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 469 – 485, 2005.

Decreasing Network Convergence Time (NCT) in response to a critical event in a network, such as a node or link failure, is significantly important for next-generation, real-time network applications. The NCT is defined as the difference in time when the critical event has occurred until the time the network is stabilized and all nodes have a loop-free path to every other node in the network domain. Network failures have been observed to be fairly common in the day to day operations of a network due to fiber cuts, hardware/software failures, faulty equipments, or router misconfigurations [1]. A link failure is usually detected immediately by lower layer protocols whereas a node failure is detected by the neighboring nodes after the expiration of a predetermined time without receiving any keep-alive message from the adjacent node. The failed node is declared down and the failure information is flooded to all nodes in the network domain. Each node, upon receiving the update, removes the failed node from its topological database and recalculates routing paths to all other nodes in its routing domain. As such, the NCT can be defined as the sum of failure detection time, information flooding time, update processing time, paths computation time and route installation time.

Guaranteeing a Quality of Service (QoS) for an application requires that the NCT of the network, where the application is running, is adequate for the operation of the application. The network is considered unstable after a node or link failure, which leads to routing failures and data packet drop, until the network has converged. During the transient period of network convergence, the application packets may be dropped due to invalid routes such as the ones transiting through the failed node or link, or the packets may get caught in routing loops. Any dropped application data packets need to be retransmitted once the network has converged. However retransmission, which is considered an acceptable option in traditional applications such as FTP, may not be feasible for the operation of most real-time applications due to an unacceptable delay incurred during retransmissions.

Emerging network technologies such as Wireless Mesh Networks (WMN) rely on routing protocols such as Open Shortest Path First (OSPF) to find optimal routes. These protocols are traditionally designed for wired networks and are mostly oblivious to the peculiar characteristics of a wireless network. Unlike wired links, the bandwidth of a wireless link can fluctuate frequently and considerably depending on variations in the radio environment. Factors such as interference, fading and shadowing affect the state of the radio link. These factors affect the stability of the network and need to be considered in the routing protocols to assure a stable and robust network operation.

In this paper, we investigate the concept of tunable failure threshold based on network congestion level. The work aims at (i) improving the NCT by reducing the failure detection time and (ii) improving the network stability in a varying network environment. Our implementation is based on OSPF routing protocol [2] and the results are based on simulations performed in *Opnet Modeler*. We analyze and compare the performance of the tunable and standard OSPF protocols in various network congestion scenarios and data traffic types.

The rest of the paper is organized as follows: In section II, we present background information on OSPF routing protocol emphasizing issues related to network convergence and present a study of the related past work done in the field of network convergence. In section III, we develop an algorithm to tune OSPF failure threshold based on the network congestion level. Section IV analyzes and compares the performance of the tunable and standard OSPF protocols. In section V, we discuss the applicability of the proposed protocol in WMN and finally in section VI, we conclude our findings.

2 OSPF Network Convergence

OSPF is the most commonly deployed link state Interior Gateway Protocol (IGP) in the current Internet infrastructure. An important characteristic of a link state protocol is that every node within a domain discovers and builds an entire view of a network topology. During initialization, an OSPF node sends hello packets to all its neighbors. The neighbors supporting the OSPF protocol recognize these packets and reply with their own OSPF packets. The neighbors then exchange their routing tables to build topological databases and are said to be adjacent. To keep the adjacency alive, the neighbors periodically send hello messages to all adjacent neighbors. Not receiving a hello packet from a particular adjacent neighbor for a predetermined fixed amount of time, called router dead interval (RDI), indicates that the neighbor is down. No state information is kept for the failed node and thus when the node recovers, the entire synchronization process is repeated.

The RDI value is a constant in the current OSPF standard and is usually set to four times the hello interval [3]. The default values for the hello interval and the RDI are 10 seconds and 40 seconds respectively. With these values, detecting a failure in a network can take up to 40 seconds. From the time of failure until the network has converged, the data traffic is continued to be forwarded to the failed node or link. The data packets, forwarded to the failed node or link, are dropped and need to be retransmitted. It has been suggested to decrease the hello and router dead intervals to expedite the failure detection time which consequently decreases the overall NCT. The values of 2 seconds for the hello interval and 8 seconds for the RDI are proposed in [4].

Routing protocol control packets usually share resources such as link bandwidth and router buffers with the data traffic. Congestion in the shared resources can thus, hinder the propagation of the control packets. The hello packets can get queued with the other data packets resulting in delayed transmission or packet drop. The adjacent neighbors will erroneously presume that the node is down if no hello packet has been received within the RDI expiration time and initiate the recovery process. Decreasing the hello and router dead intervals increases the probability of losing adjacencies and declaring spurious failures. As such, the RDI value is an important parameter in an OSPF network configuration. A small RDI value expedites real failure detections but may result in spurious failure decisions due to missed or delayed hello packets in a congested network whereas a large RDI value prolongs the real failure detection time.

There has been a growing interest in decreasing the NCT and improving network stability, and several techniques have been proposed in some recent publications. Choudhury et al. [5] explores the possibility of differentiated queuing that gives the hello control packets higher priority over other control and data packets. This proposal is backed by simulation results that show significant improvement in the network stability and scalability. The idea of keeping backup routes is investigated in [6] which proposes replacing slow path re-computation with faster path switching in case of failure. This proposal is more beneficial for link failures that can be detected almost immediately and the switching can be made fast enough to make the failure oblivious from the upper layers. Narvaez et al. [7] and Wu et al. [8] propose minimizing the flooding domain by performing localized restoration and running incremental shortest path first (SPF) algorithm. The unidirectional restoration algorithm is documented in [7] whereas [8] extends the algorithm for bidirectional links. Villamizar [9] proposes periodically caching the SPF results and using the cached information in case of failure and recovery for topology rediscovery. Caching allows transmitting minimal information for synchronization as only the updates that occurred since the last caching are required to be transmitted. The idea of reducing the hello interval to expedite the failure detections and its implications on network stability is investigated in [10]. The paper concludes that the selection of an optimal hello interval value is strongly influenced by the network congestion and the number of links in the topology.

The trade-off between reducing the RDI value to expedite the failure detection time and its effect on network stability is investigated in [11]. It is shown through simulations with constant bit-rate (CBR) and FTP data traffic that the optimal RDI value that lowers the failure detection time while assures a stable routing operation depends on the network congestion level. For improved network performance, [11] proposes tuning the RDI value based on the network congestion level, which is discussed in this paper.

3 State-Based Tunable Failure Threshold

In this section, we describe an algorithm to dynamically tune the RDI value based on network congestion level for a point-to-point OSPF network. The OSPF standard requires that the hello and router dead intervals must be the same on both ends of the point-to-point link and thus any change in the RDI value must be propagated and configured at both interfaces. Furthermore, we need to ensure that the algorithm can incorporate asymmetric traffic flows in opposite directions. To facilitate the consistency, we introduce the notion of master and slave in the connecting interfaces. The designation can be made simply by choosing the interface with a higher IP address as the master. Any update in the RDI value will be done only by the master interface. A slave, on the other hand, can suggest a new RDI value to the master and if the request is accepted, the new value is propagated back to the slave.

The steps involved in the tunable RDI algorithm are discussed in detail in the following sections:

3.1 Detecting Network Congestion

To determine the network congestion level, we maintain a history of hello packet arrivals at each interface. The frequency of updating the RDI value needs to be such that any short-lived traffic variations are ignored by the algorithm to minimize frequent changes in the RDI value and facilitate the network stability. On the other hand, the performance of the algorithm is measured by its response time which is the difference in time when the network congestion level has changed until the time the RDI value is updated to reflect the change in the network congestion. The optimal RDI update time that lowers the algorithm response time while assures the network stability depends on the traffic characteristics of the network [11]. In our algorithm, we determine the network congestion and update the RDI value on the arrival of every twentieth hello packet. Based on the history information, we determine the number of hello packets sent by the sending interface and calculate the hello packet drop rate. i.e.

$$\text{Number of hello packets received: } H_r = 20 \tag{1}$$

$$\text{Number of hello packets sent: } H_s = (t_2 - t_1) \text{ / hello interval} \tag{2}$$

$$\text{Hello packet drop rate} = 1 - (H_r / H_s) \tag{3}$$

Where t_1 corresponds to the time when the first hello packet in the current sequence of twenty hello packets is received and t_2 corresponds to the arrival time of the twentieth hello packet. We divide the network congestion into five states (no congestion, low congestion, medium congestion, high congestion and very high congestion) based on the hello packet drop rate. Upon arrival of every twentieth hello packet at an interface, we calculate the hello packet drop rate and determine the desired RDI value from the look-up table given in Table 1 [11]. The current RDI value is then compared with the desired RDI value and the current value is incremented or decremented, as and if required, by one hello interval. These desired RDI values act as upper and lower bounds to prevent the RDI value grow or drop unboundedly. This approach of maintaining a history of hello packets to determine the network congestion level and updating the RDI value by one hello interval at a time circumvents abrupt changes in the RDI value due to sudden traffic burst or short-lived traffic variations and thus, facilitates the network stability.

Table 1. Network Congestion Lookup List

Hello Packet Drop Rate	Network Congestion State	Desired RDI
[0, 0.1]	No Congestion	6
(0.1, 0.25]	Low Congestion	8
(0.25, 0.5]	Medium Congestion	12
(0.5, 0.75]	High Congestion	16
(0.75, 1]	Very High Congestion	20

3.2 Propagating RDI Values

The hello packets are used for the propagation of new RDI values. An OSPF hello packet contains the hello interval and the RDI of the originating interface. In the current OSPF standard, these values contained in the hello packet must match the values stored locally at the receiving interface and the hello packet is discarded in case of any discrepancy. In the tunable RDI algorithm, any new RDI value from the master or suggested RDI value from the slave is contained in the hello packet. The receiving interface recognizes that the RDI contained in the packet is different from the local RDI and runs the tunable RDI algorithm. This approach of using the hello packets for the propagation of new RDI values allows the implementation of the tunable RDI algorithm without modifying the existing OSPF packet formats or adding any extra communication overhead.

3.3 Updating RDI Values

An increase in the RDI value requires signaling from only one interface to ensure that the RDI is increased when either interface is experiencing congestion. If the master interface determines that the local RDI value is less than the desired value, the master increments the local RDI value by one hello interval and sends the new value to the slave in the next hello packet. The slave updates its local RDI value upon receiving that hello packet. If the slave is experiencing congestion and wants to increase the RDI value, the slave sends a RDI value incremented by one hello interval in the next hello packet destined for the master. The slave however, does not modify its local RDI value. The master upon receiving the hello packet realizes that the slave is requesting to increase the RDI value and accepts the request unconditionally. The master then increments the local RDI value by one hello interval and sends the new value to the slave in the next hello packet resulting in the slave updating its RDI value.

Decreasing the RDI value requires acceptance from both interfaces to ensure that the RDI value is decreased only if both interfaces are experiencing low congestion. As such, only one interface needs to initiate the request while the other interface, upon receiving the request, determines if it is feasible to decrease the RDI value. In our implementation, the slave initiates the decrease RDI request. When the slave determines that the current RDI value is higher than the desired one, the slave sends a RDI value decremented by one hello interval in the next hello packet destined for the master. Upon receiving the request, the master consults its history to determine the possibility of reducing the RDI value. If the request is approved, the master lowers its local RDI value by one hello interval and sends the new value to the slave in the next hello packet resulting in the slave reducing its RDI value.

If the hello packet containing the slave suggestion is dropped, the slave will never find out that the master has not received the suggestion. The slave will presume that the request is rejected by the master. To handle this situation in our implementation, the slave sends the suggested RDI value to the master in up to three consecutive hello packets and if still the slave has not received the desired RDI value from the master, the request is presumed to be rejected.

The pseudo code for the tunable RDI algorithm when a hello packet is received at an interface is given in Fig. 1.

```
Event: hello packet received

/* Parse hello packet and extract RDI value from it (msg.rdi) */

If interface.rdi ≠ msg.rdi              /* msg.rdi differs from the locally stored RDI value */
    If interface is a slave
       interface.rdi = msg.rdi                   /* Slave accepts new value from the master */
    Else if interface is a master                /* msg.rdi is a suggestion from the slave */
       If msg.rdi > interface.rdi                /* Suggestion is to increase the RDI value */
          interface.rdi = msg.rdi                /* Always accept the increase RDI request */
       Else                                      /* Suggestion is to decrease the RDI value */
      /* Determine the desired RDI value for the current congestion level from lookup table */
          If desired_rdi < interface.rdi         /* Congestion level has decreased */
            interface.rdi = msg.rdi              /* Accept the decrease RDI request */
        End if
      End if
    End if
End if



/* Check if the RDI value can be increased or decreased. The adjacent interface will receive any
update in the next hello packet */

/* Calculate the hello packet drop rate and find the desired RDI value from the lookup table */
If interface.rdi < desired_rdi                   /* Congestion level has increased */
    If interface is a master                     /* Master increases its RDI value */
       interface.rdi = interface.rdi + interface.hello_interval
    Else if interface is a slave                 /* Slave suggests a higher RDI value */
       interface.suggested_rdi = interface.rdi + interface.hello_interval
    End if
Else if interface.rdi > desired_rdi              /* Congestion level has decreased */
    If interface is a slave                      /* Slave suggests a lower RDI value */
       interface.suggested_rdi = interface.rdi – interface.hello_interval
    End if
End if
```

Fig. 1. Tunable RDI OSPF Algorithm Pseudo Code

4 Performance Evaluation

In this section, we analyze and compare the performance of the tunable and standard OSPF protocols. We developed two sets of test cases: (i) simulated hello packet drop and (ii) congestion-based hello packet drop. In the former case, the hello packets arriving at an interface are explicitly dropped with a known probability to simulate the network congestion. The simulated hello packet drop test cases allow us understand the dynamics of the algorithm in a controlled environment. In the latter case, we investigate the performance of the tunable and standard OSPF protocols in the presence of data traffic when both the data and control traffic flows are assigned equal priority. The congestion-based hello packet drop test cases give us an insight to the behavior of the applications in a network running the tunable OSPF routing protocol. To compare the performance of the protocols, we employ the method of *'Correlated Sampling'* in which the same set of random numbers is used to simulate both systems [12]. For each test case, we ran 30 replications and report the results with 95% confidence interval (CI).

The performance of the tunable and standards OSPF protocols is compared in terms of the number of adjacencies lost in each case. Since no real node or link failure is simulated in the network, all adjacency losses are spurious and thus, a lower number depicts superior performance. Each simulation is run for 1500 seconds and batch size is set to 100 seconds i.e. for every 100 seconds, we count the number of adjacencies lost in that particular batch. The hello interval and the RDI are configured to 2 seconds and 8 seconds respectively as recommended in [4] to expedite real failure detections in a network.

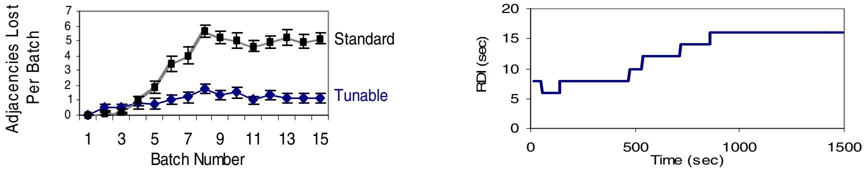
4.1 Simulated Hello Packet Drop Tests

In the simulated hello packet drop cases, we explicitly drop a predefined percentage of hello packets arriving at an interface to simulate network congestion for a two-node point-to-point OSPF network in the absence of any data traffic. As each link in a point-to-point OSPF network is independent, this simple topology is deemed sufficient for simulations in a controlled test environment.

4.1.1 Gradually Increasing Network Congestion

In this test case, we start the simulation with no congestion (0% packet drop) and increase the hello packet drop rate by 10% every 100 seconds on one interface until it reaches 70% at time=700 seconds. The congestion is then kept at 70% until the end of simulation. The hello packet drop rate at the other interface is kept at 0% throughout the simulation.

Fig. 2(a) shows the number of adjacencies lost per batch for the tunable and standard OSPF protocols. We notice that initially until batch 3, the tunable OSPF resulted in a slightly higher adjacency loss rate per batch; however after batch 3, the standard OSPF resulted in a much higher adjacency loss rate. The reason for the initial slightly higher loss rate in the tunable OSPF is that, in the beginning of the simulation, the hello packet drop rate of 0% allowed the tunable OSPF to reduce the RDI value from 8 to 6 seconds as shown in the RDI graph of Fig. 2(b). As such, when the congestion developed, the tunable OSPF declared adjacency down after 3 consecutive hello packet drops as compared to 4 consecutive hello packet drops in the standard OSPF protocol. However, the tunable RDI algorithm responded to the increasing hello packet drop rate and gradually increased the RDI value and thus, resulted in lower adjacency loss rate per batch later in the simulation. The standard OSPF is oblivious to the network congestion and continued to lose adjacencies at a higher rate until the end of the simulation.



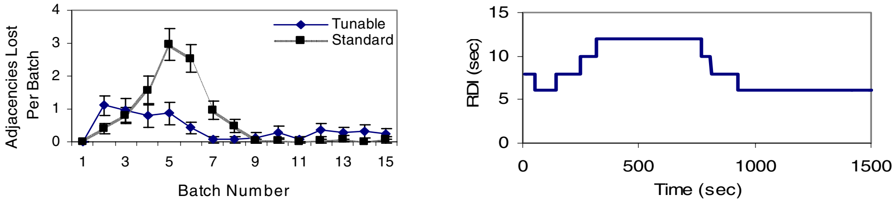
(a) Number of adjacencies lost (95% CI)

(b) RDI graph for tunable OSPF

Fig. 2. Simulated Hello Packet Drop Test Case 1 (Gradually increasing network congestion)

4.1.2 Gradually Increasing then Decreasing Congestion

In this test case, we gradually build congestion on one interface, starting at 20% at time=100 seconds and increasing the hello packet drop rate by 10% every 100 seconds until the congestion reaches 50% at time=400 seconds. The congestion is then decreased by 10% every 100 seconds until it reaches 10% at time=800 seconds. The congestion is then kept at 10% until the end of simulation. The hello packet drop rate at the other interface is kept at 0% throughout the simulation.



(a) Number of adjacencies lost (95% CI)

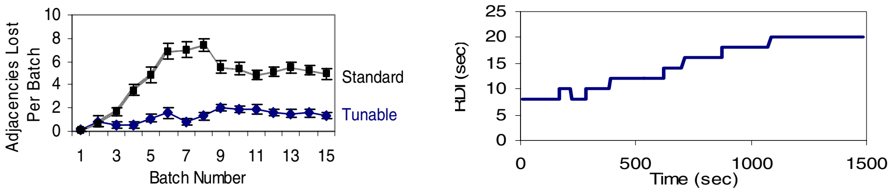
(b) RDI graph for tunable OSPF

Fig. 3. Simulated Hello Packet Drop Test Case 2 (Gradually increasing then decreasing network congestion)

Fig. 3(a) shows a graph of the number of lost adjacencies per batch by the tunable and standard OSPF protocols. We notice that the standard OSPF results in higher adjacency loss rate in the first half whereas the tunable OSPF resulted in a slightly higher adjacency loss rate in the latter half of the simulation. The reason for the first half has already been explained in the first test case. For the latter half, the hello packet drop rate of 10% allowed the tunable OSPF to decrease the RDI to 6 seconds as shown in the RDI graph of Fig. 3(b). With the RDI of 8 seconds for the standard OSPF, the tunable OSPF has a higher probability of declaring the adjacencies down in response to hello packets drop as compared to the standard OSPF protocol. However, on the other hand, the tunable OSPF protocol will detect a real failure in 6 seconds as opposed to 8 seconds by the standard OSPF and thus, decrease the overall network convergence time.

4.1.3 Asymmetric Network Congestion

In this test case, we start with no congestion and increase the hello packet drop by 10% every 100 seconds on both interfaces. However after time=600 seconds, the packet drop at one interface is decreased by 10% every 100 seconds until the hello packet drop rate at that interface reaches 10% at time=900 seconds and then is kept at that level. On the other interface, the hello packet drop rate keeps increasing by 10% every 100 seconds until the congestion level is raised to 80% at time=800 seconds and then is kept at that level. The intent of this test case is to analyze the algorithm for asymmetric traffic flows. As the interface where the hello packet drop rate is decreasing will request to decrease the RDI value, the interface with high congestion will reject the decrease RDI request and keep the RDI value high.



(a) Number of adjacencies lost (95% CI)

(b) RDI graph for tunable OSPF

Fig. 4. Simulated Hello Packet Drop Test Case 3 (Asymmetric network congestion)

Fig. 4(a) shows the number of lost adjacencies per batch and Fig. 4(b) shows the corresponding RDI graph for the tunable RDI case. It is evident that the RDI is increased gradually and is kept high until the end of simulation. The interface with decreasing network congestion periodically requests to decrease the RDI value but the request is rejected by the interface with persistent high network congestion. The adjacencies loss rate graph depicts the same pattern as in the first test case and has already been explained.

4.2 Congestion-Based Hello Packet Drop Tests

In the congestion-based hello packet drop test cases, we analyze the performance of the standard and tunable OSPF protocols when the network resources are shared between the data and control traffic. We experimented with UDP-based CBR and TCP-based FTP data traffic flows.

4.2.1 CBR Data Traffic

In this section, we analyze the performance of the standard and tunable OSPF protocols in the presence of CBR data traffic. The CBR traffic is unreliable and unidirectional. It is predictable as the packet size, packet interval and stream duration are known. Most audio and video streams are based on UDP and thus, are also unreliable and unidirectional. Due to the absence of flow control in UDP, the source transmits the traffic at a constant and predictable rate. Due to these similarities, we can approximate the real-time UDP streaming traffic with the CBR traffic.

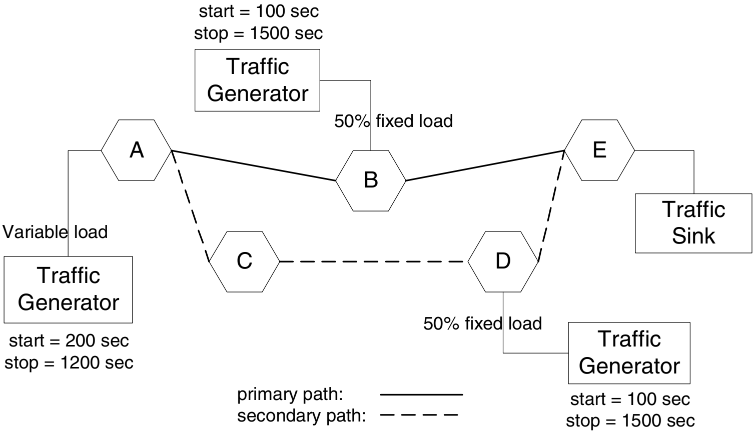
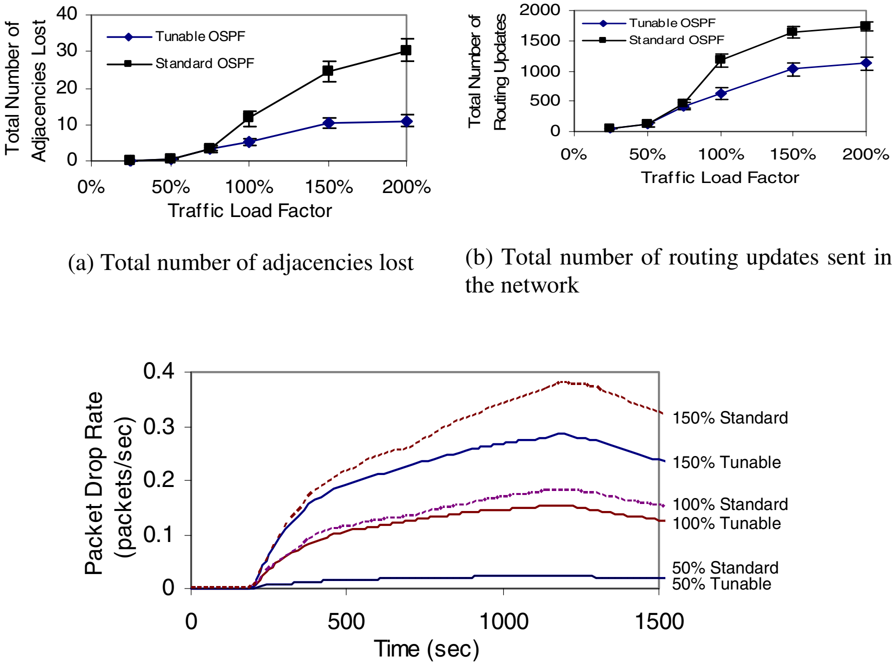


Fig. 5. Simulation Network Topology for Congestion-based Hello Packet Drop Test Case with CBR Data Traffic

The network topology is shown in Fig. 5. The links connecting the routers are PPP-DS3 supporting a maximum bandwidth of 44.7Mbits/second. Routers ‘B’ and ‘D’ are connecting to traffic sources that are generating data traffic at a constant rate of 22Mbits/second (50% of the link bandwidth) from time=100 seconds to time=1500 seconds in all the simulation scenarios. The CBR traffic generated by a source connected to router ‘A’ is varied during the experiment and we observe the performance of the tunable and standard OSPF protocols when all traffic flows are destined for router ‘E’. As both the data and control packets are assigned equal priority and thus share the router FIFO queue, increasing the data traffic results in the control traffic being delayed or dropped. Under normal conditions, the data traffic originating from router ‘A’ takes path ABE which we refer to from here on as the primary path. However, as we increase the traffic rate from router ‘A’, congestion develops at router ‘B’ resulting in both the control and data packets being delayed and dropped. Not receiving a hello packet within the RDI from router ‘B’ results in router ‘E’ advertising router ‘B’ down, and all nodes are notified about the router ‘B’ spurious failure. Router ‘A’, upon receiving the update, re-computes its routing table resulting in redirecting the data traffic to path ACDE. The re-routing gradually alleviates the congestion at router ‘B’ and eventually a hello packet reaches router ‘E’ resulting in the rediscovery of router ‘B’. The rediscovery of router ‘B’ results in redirecting the router ‘A’ data traffic back to the primary path. Frequent rerouting causes network instability and data packets drop. Each update (failure or rediscovery) is flooded to all nodes in the network domain resulting in excessive bandwidth usage as well as inconsistent network view and excessive CPU usage at the node level.



(a) Total number of adjacencies lost

(b) Total number of routing updates sent in the network

(c) Average packet drop rate for selected loads

Fig. 6. Comparison of Tunable and Standard OSPF Protocols Performance with CBR data Traffic (buffer size=1K bytes, packet size=64 bytes, 30 replications, 95% CI)

Fig. 6 shows the results of overloading traffic on the standard and tunable OSPF protocols when the traffic source connected to router ‘A’ is generating a CBR data traffic of 25%, 50%, 75%, 100%, 150% and 200% of the link bandwidth. Fig. 6(a) and Fig. 6(b) compare the number of adjacencies lost and the number of routing updates sent in the network respectively. Up to 50% traffic load, there is no congestion at router ‘B’ and thus, the two protocols behave comparably. However, the performance of the standard OSPF protocol deteriorates considerably as the traffic load is increased beyond 50%, due to the congestion at router ‘B’. The tunable OSPF protocol resulted in about 3 times lesser number of adjacency losses as compared to the standard OSPF protocol when the traffic load is increased beyond 150%. Also, it is evident that the number of adjacency losses in the standard OSPF protocol increased linearly with the increase in the traffic load. The tunable OSPF, however, is adaptable to the applied traffic load and thus, the number of adjacency losses by the tunable OSPF protocol is almost the same for both the 150% and 200% traffic load cases. Each spurious failure and rediscovery is flooded in the network as an update and thus, as expected, the number of routing updates increases with the increase in the number of lost adjacencies as shown in Fig. 6(b).

To demonstrate the effect of number of spurious lost adjacencies on the network stability, Fig. 6(c) shows average packet drop rate for selected traffic loads with the tunable and standard OSPF protocols. The two protocols behave comparably for 50% traffic load whereas the standard OSPF dropped 15% and 25% more packets as compared to the tunable OSPF for 100% and 150% loads respectively, due to the spurious lost adjacencies and frequent re-routing, causing routing failures and network instability.

4.2.2 FTP Data Traffic

In this section, we compare the performance of the standard and tunable OSPF with heavy FTP traffic load. The simulation network topology is shown in Fig. 7. The users in LAN1 and LAN2 request files located at the FTP server. The requests from LAN1 start at time=200 seconds and continue until time=1300 seconds whereas the users in LAN2 requests files from time=500 seconds until time=1000 seconds. The size of each file located at the server is 500K bytes whereas the user requests are made with an inter-arrival time distributed exponentially with λ=50.

Fig. 8 compares the performance of the standard and tunable OSPF protocols with the heavy FTP traffic load. We see that the tunable OSPF protocol resulted in a total of 3 lost adjacencies as compared to 9 adjacency losses by the standard OSPF protocol. In the standard OSPF protocol, most of the adjacencies are lost between time=500 seconds to time=1000 seconds when the users in LAN1 and LAN2 are making simultaneous requests. Both the standard and tunable OSPF protocols lost their first adjacency at time=600 seconds due to increase in the network congestion. However, the tunable OSPF lost its second adjacency at time=1270 seconds. The standard OSPF protocol lost 5 more adjacencies until time=1000 seconds. The reason for the difference is that the tunable OSPF protocol algorithm adjusted to the increase in the network congestion level and accordingly increased the RDI value. As such, there are no more adjacency losses until the high congestion level persisted in the network. The standard OSPF protocol is oblivious to the network congestion and continued losing adjacencies at regular intervals.

After the users in LAN2 stopped making the file requests at time=1000 seconds, the network congestion level started decreasing. From time=1000 seconds to time=1500 seconds, the standard OSPF protocol lost 3 more adjacencies as opposed to 2 more adjacencies by the tunable OSPF protocol. The adjacency losses by the tunable OSPF protocol are caused by the varying network congestion level. With the decreasing network congestion level, the tunable RDI algorithm gradually decreases the RDI value. As the RDI value is decreased, it becomes more likely to lose the adjacencies due to delayed or dropped hello packets.

When we compare the IP traffic received (packets/seconds) by the users in LAN1 and LAN2 with the standard and tunable OSPF protocol as shown in Fig. 8(b) and

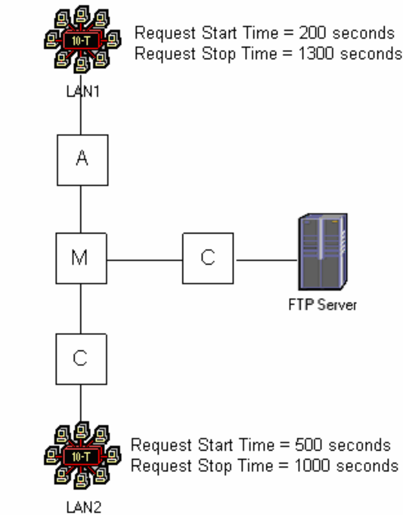
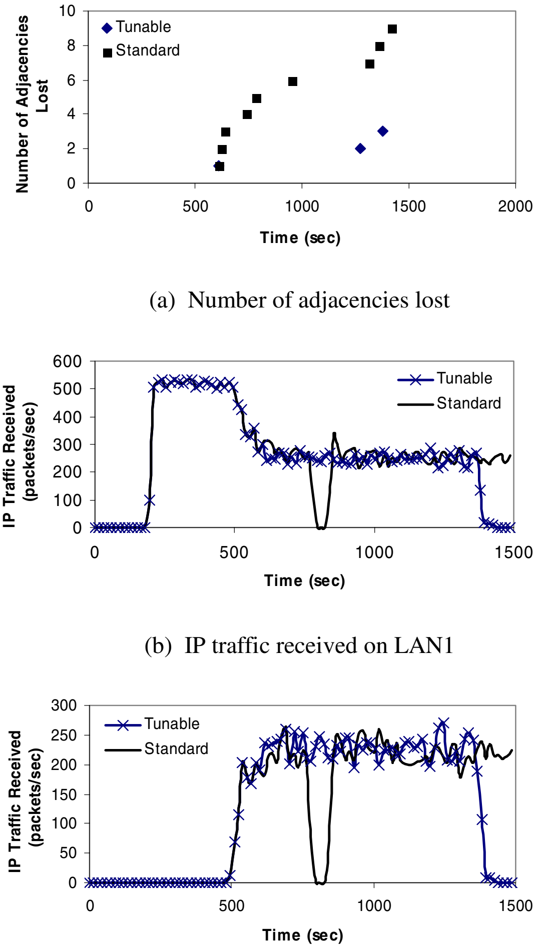

10-T
Request Start Time = 200 seconds
Request Stop Time = 1300 seconds
LAN1
A
M
C
FTP Server
C
10-T
Request Start Time = 500 seconds
Request Stop Time = 1000 seconds
LAN2

Fig. 7. Simulation Network Topology for Congestion-based Hello Packet Drop Test Case with Heavy FTP Data Traffic

Fig. 8(c) respectively, we notice that the tunable OSPF protocol provides a more stable network operation. Due to 6 adjacency losses by the standard OSPF protocol between time=500 seconds until time=1000 seconds, we notice a sudden drop in the IP traffic received by both LAN1 and LAN2 at around time=800 seconds. The reason for the drop is that losing a number of adjacencies in a short interval results in inconsistent network view at the nodes which leads to routing failures and consequently data traffic being dropped. There is no such drop in the received traffic when the tunable OSPF is the employed routing protocol. As such, the deployment of the tunable OSPF protocol facilitates a more stable network environment.

When we analyze the IP traffic received pattern from time=1000 seconds to time=1500 seconds, we note that both protocols are showing comparable performance. However we see that in the case of the tunable OSPF protocol, the file transfer finished at time=1425 seconds whereas the file transfer with the standard OSPF protocol continued until the end of simulation at time=1500 seconds. The reason for this difference is that the tunable OSPF protocol provided a more stable network environment with no interruptions and routing failures. As such, the file transfer completed early with the tunable OSPF protocol as compared to the standard OSPF protocol where the routing failures resulted in dropped packets. The dropped data packets need to be retransmitted after the network has stabilized which prolonged the overall file transfer time.



(a) Number of adjacencies lost

(b) IP traffic received on LAN1

(c) IP traffic received on LAN2

Fig. 8. Comparison of Tunable and Standard OSPF protocols with Heavy FTP Traffic (File size=500K bytes, λ=50)

5 Applicability of Tunable OSPF Protocol in WMN

Wireless mesh networks (WMN) is an emerging broadband Internet access technology that is drawing significant attention [13]. The WMN backhaul consists of fixed 802.11 wireless nodes connected to each other via point-to-point radio links as shown in Fig. 9. The mesh network ensures the availability of multiple paths for each node in the network and relies on routing protocols, such as OSPF, to find an optimum route. Unlike wired links, the bandwidth on a wireless link can fluctuate frequently and considerably due to variations in the radio environment. To keep an acceptable signal-to-noise ratio (SNR), most WMN supports multi-rate bandwidth [14]. As the radio link quality deteriorates, the supported bandwidth is decreased to reduce the transmission rate, resulting in a decreased packet error rate. The decrease in bandwidth results in increasing the network congestion level and raises the probability of dropping the hello packets and losing the adjacencies due to spurious failure decisions. The standard OSPF protocol, which is oblivious to the network congestion, starts declaring adjacencies down after the expiration of the fixed RDI. The tunable OSPF protocol, on the other hand, adapts to the varying network congestion level. As the network congestion increases due to the decrease in the link bandwidth, the tunable OSPF protocol increases the RDI value to avoid the spurious failures and facilitates network stability. As the radio link quality improves, the link bandwidth increases resulting in a decrease in the network congestion level. The tunable OSPF protocol gradually decreases the RDI value to expedite the failure detection time and thus, reduces the overall NCT.

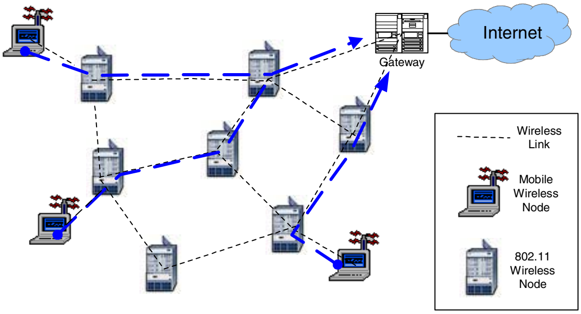


Fig. 9. Wireless Mesh Networks

6 Conclusion

This paper shows that network convergence time and number of spurious lost adjacencies in an OSPF network can be reduced by tuning the router dead interval (RDI) based on network congestion level. An algorithm is proposed for a point-to-point OSPF network to dynamically tune the RDI value based on the network congestion level. Results obtained from simulations of the proposed algorithm showed reduced numbers of spurious adjacency losses under high network congestion as compared to standard OSPF while improving the real failure detection time in low network congestion. The algorithm may lead to higher network convergence time under heavy congestion but improves the stability of the network.

We compared the performance of the tunable and standard OSPF protocols with CBR data traffic which closely models the real-time data traffic and with FTP data traffic. The tunable OSPF protocol has shown to provide a more stable network environment with uninterrupted data transfer and facilitated shorter download times. Furthermore, it is intuitively explained that the performance and stability of the emerging Wireless Mesh Networks (WMN), which is subject to radio channel anomalies, can be improved by employing the proposed tunable OSPF protocol.

References

1. G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002
2. J. Moy, "OSPF Version 2," RFC 2328, April 1998
3. Cisco Systems, "OSPF design guide," [online], http://www.cisco.com/warp/public/104/1.html
4. C. Alaettinoglu, V. Jacobson and H. Yu, "Towards millisecond IGP convergence," [online], http://www.nanog.org/mtg-0010/ppt/cengiz.pdf , NANOG 20, October 2000
5. G. Choudhury, V. Sapozhnikov, G. Ash, A. Maunder and V. Manral, "Prioritized treatment of specific OSPF version 2 packets and congestion avoidance," Internet Draft, draft-ietf-ospf-scalability-09.txt, December 2004
6. J. Pu, E. Manning and G. C. Shoja, "Routing reliability analysis of partially disjoint paths," in *IEEE Conference on Communications, Computers, and Signal Processing*, 2001, Vol. 1, pp. 79-82, August 2001
7. P. Narvaez, K. Siu and H. Tzeng, "Local restoration algorithm for link-state routing protocols," in *Eight International Conference on Computer Communications and Networks*, pp. 352-357, 1999
8. J. Wu, X. Lin, J. Cao and J. Weijia, "An extended fault-tolerant link-state routing protocol in the Internet," in *Eighth Internal Conference on Parallel and Distributed Systems*, pp. 331-337, 2001
9. C. Villamizar, "Convergence and restoration techniques for ISP interior routing," [online], http://www.nanog.org/mtg-0206/ppt/curtis.pdf
10. M. Goyal, K. Ramakrishnan, and W. Feng, "Achieving faster failure detection in OSPF networks," *Proceedings of ICC 2003*, Vol. 1, pp. 296-300, May 2003
11. Siddiqi, "Improving Network Convergence Time and Network Stability of an OSPF-Routed IP Network," Master's Dissertation, Ottawa-Carleton Institute for Electrical and Computer Engineering, Carleton University, Ottawa 2004
12. J. Banks, J. Carson, B. Nelson and D. Nicol, *Discrete-event system simulation.* Prentice Hall, pp. 456-467, 2000
13. J. Jun and M. Sichitiu, "The nominal capacity of wireless mesh networks," *IEEE Wireless Communications*, Vol. 10, pp. 8-14, October 2003
14. Acharya and A. Misra, "High-performance architectures for IP-based multihop 802.11 networks," *IEEE Wireless Communications*, Vol. 10, pp. 22-28, October 2003

Non-cooperative Forwarding in Ad-Hoc Networks

Eitan Altman[1], Arzad A. Kherani[1], Pietro Michiardi[1], and Refik Molva[2]

[1] INRIA, 06902 Sophia Antipolis, France
{Eitan.Altman, alam}@sophia.inria.fr
[2] GET/EURECOM, Sophia-Antipolis, France
{Pietro.Michiardi, molva}@eurecom.fr

Abstract. A wireless Ad-hoc network is expected to be made up of energy aware entities (nodes) interested in their own perceived performance. An important problem in such a scenario is to provide incentives for collaboration among the participating entities. Forwarding packets of other nodes is an example of activity that requires such a collaboration. However, it may not be in interest of a node to always forward the requesting packets. At the same time, not forwarding any packet may adversly affect the network functioning. Assuming that the nodes are rational, i.e., their actions are strictly determined by their self-interest, we view the problem in framework of non-cooperative game theory and provide a simple punishing mechanism considering end-to-end performance objectives of the nodes. We also provide a distributed implementation of the proposed mechanism. This implementation has a small computational and storage complexity hence is suitable for the scenario under consideration.

Keywords: Game theory, Stochastic approximation algorithm.

1 Introduction

In order to maintain connectivity in an Ad-hoc network, mobile terminals should not only spend their resources (battery power) to send their own packets, but also for forwarding packets of other mobiles. Since Ad-hoc networks do not have a centralized base-station that coordinates between them, an important question that has been addressed is to know whether we may indeed expect mobiles to collaborate in such forwarding. If mobiles behave selfishly, they might not be interested in spending their precious transmission power in forwarding of other mobile's traffic. A natural framework to study this problem is noncooperative game theory. As already observed in many papers that consider noncooperative behavior in Ad-hoc networks, if we restrict to simplistic policies in which each mobile determines a fixed probability of forwarding a packet, then this gives rise to the most "aggressive" equilibrium in which no one forwards packets, see e.g. [3–Corollary 1], [4], thus preventing the system to behave as a connected network. The phenomenon of aggressive equilibrium that severely affects performance has

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 486–498, 2005.

also been reported in other noncooperative problems in networking, see e.g. [1] for a flow control context (in which the aggressive equilibrium corresponds to all users sending at their maximum rate).

In order to avoid very aggressive equilibria, we propose strategies based on threats of punishments for misbehaving aggressive mobiles, which is in the spirit of a well established design approach for promoting cooperation in Ad-hoc networks, carried on in many previous works [3, 7]. In all these references, the well known "TIT-FOR-TAT" (TFT) strategy was proposed. This is a strategy in which when a misbehaving node is detected then the reaction of other mobiles is to stop completely forwarding packets during some time; it thus prescribes a threat for very "aggressive" punishment, resulting in an enforcement of a fully cooperative equilibrium in which all mobiles forward all packets they receive (see e.g. [3–Corollary 2]). The authors of [6] also propose use of a variant of TFT in a similar context.

In this work we consider a less agressive punishment policy. We simply assume that if the fraction q' of packets forwarded by a mobile is less than the fraction q forwarded by other mobiles, then this will result in a decrease of the forwarding probability of the other mobiles to the value q'. We shall show that this will indeed lead to non-aggressive equilibria, yet not necessarily to complete cooperation. The reasons for adopting this milder punishment strategy are the following:

1. There has been criticism in the game-theoretical community on the use of aggressive punishments. For example, threats for aggressive punishments have been argued not to be credible threats when the punishing agent may itself loose at the punishing phase. This motivated equilibria based on more credible punishments known as subgame perfect equilibria [5].
2. An individual that adopts an "partially-cooperative" behavior (i.e. forwards packets with probability $0 < q < 1$) need not be considered as an "aggressive" individual, and thus the punishment needs not be "aggressive" either; it is *fair* to respond to such a partially-cooperative behavior with a partially-cooperative reaction, which gives rise to our mild punishment scheme.
3. The TFT policy would lead to complete cooperation at equilibrium. However, our milder punishment seems to us more descriptive of actual behavior in the society in which we do not obtain full cooperation at equilibrium (for example in the behavior of drivers on the road, in the rate of criminality etc.) It may indeed be expected that some degree of non-cooperative behavior by a small number of persons could result in larger and larger portions of the society to react by adopting such a behavior.

As already mentioned, incentive for cooperation in Ad-hoc networks have been studied in several papers, see [3, 4, 6, 7]. Almost all previous papers however only considered utilities related to successful transmission of a mobile's packet to its neighbor. In practice, however, multihop routes may be required for a packet to reach its destination, so the utility corresponding to successful transmission depends on the forwarding behavior of all mobiles along the path. The goal of our paper is therefore to study the forwarding taking into account the multihop topological characteristics of the path.

Most close to our work is the paper [3] which considers a model similar to ours (introduced in Section 2 below). [3] provides sufficient condition on the network topology under which each node employing the "aggressive" TFT punishment strategy results in a Nash equilibrium. In the present paper, we show that a less aggressive punishment mechanism can also lead to a Nash equilibrium which has a desirable feature that it is less resource consuming in the sense that a node need not accept all the forwarding request. We also provide some results describing the structure of the Nash equilibrium thus obtained (Section 5). We then provide a distributed algorithm which can be used by the nodes to compute their equlibrium strategies and enforce the punishment mechanism using only local information (Section 6). The algorithm is implemented using the MATLAB suite and some numerical results are presented (Section 7). Scetion 8 concludes the paper.

2 The Model

Consider an Ad-hoc network described by a directed graph $G = (N, V)$. Along with that network, we consider a set of source-destination pairs O and a given routing between each source s and its corresponding destination d, of the form $\pi(s,d) = (s, n_1, n_2, \ldots, n_k, d)$, where $k = k(s,d)$ is the number of intermediate hops and $n_j = n_j(s,d)$ is the jth intermediate node on path $\pi(s,d)$. We assume that mobile j forwards packets (independently from the source of the packet) with a fixed probabilty γ_j. Let $\underline{\gamma}$ be the vector of forwarding probabilities of all mobiles. We assume however that each source s forwards its own packets with probability one. For a given path $\pi(s,d)$, the probability that a transmitted packet reaches its destination is thus: $p(s,d;\underline{\gamma}) = \prod_{j=1}^{k(s,d)} \gamma(n_j(s,d))$.

If i belongs to a path $\pi(s,d)$ we write $i \in \pi(s,d)$. For a given path $\pi(s,d)$ of the form $(s, n_1, n_2, \ldots, n_k, d)$ and a given mobile $n_j \in \pi(s,d)$, define the set of intermediate nodes before n_j to be the set $S(s,d;n_j) = (n_1, ..., n_{j-1})$. The probability that some node $i \in \pi(s,d)$ receives a packet originating from s with d as its destination is then given by $p(s,d;i,\underline{\gamma}) = \prod_{j \in S(s,d;i)} \gamma(j)$.

Note that $p(s,d;d,\underline{\gamma}) = p(s,d;\underline{\gamma})$, the probability that node d receives a packet originating from source s and having d as its destination.

Define $O(i)$ to be all the paths in which a mobile i is an intermediate node. Let the rate at which source s creates packets for destination d be given by some constant λ_{sd}. Then the rate at which packets arrive at node i in order to be forwarded there is given by $\xi_i(\underline{\gamma}) = \sum_{\pi(s,d) \in O(i)} \lambda_{sd} p(s,d;i,\underline{\gamma})$.

Let E_f be the total energy needed for forwarding a packet (which includes the energy for its reception and its transmission). Then the utility of mobile i that we consider is

$$U_i(\underline{\gamma}) = \sum_{n:(i,n) \in O} \lambda_{in} f_i(p(i,n;\underline{\gamma})) + \sum_{n:(n,i) \in O} \lambda_{ni} g_i(p(n,i;\underline{\gamma})) - aE_f \xi_i(\underline{\gamma}), \quad (1)$$

where f_i and g_i are utility functions that depend on the success probabilities associated with node i as a source and as a destination respectively and a is some multiplicative constant. We assume that $f_i(\cdot)$ and $g_i(\cdot)$ are nondecreasing concave in their arguments. The objective of mobile i is to choose γ_i that maximizes $U_i(\underline{\gamma})$. We remark here that similar utility function is also considered in [3] with the difference that node's utility does not include its reward as a destination, i.e., they assume that $g_i(\cdot) \equiv 0$.

Definition: *For any choices of strategy $\underline{\gamma}$ for all mobiles, define $(\gamma_i', \underline{\gamma}^{-i})$ to be the strategy obtained when only player i deviates from γ_i to γ_i' and other mobiles maintain their strategies fixed.*

In a noncooperative framework, the solution concept of the optimization problem faced by all players is the following:

Definition: *A Nash equilibrium, is some strategy set $\underline{\gamma}^*$ for all mobiles such that for each mobile i,*

$$U_i(\underline{\gamma}^*) = \max_{\gamma_i'} U_i(\gamma_i', (\underline{\gamma}^*)^{-i}).$$

We call $argmax_{\gamma_i'} U_i(\gamma_i', \underline{\gamma}^{-i})$ the set of optimal responses of player i against other mobiles policy $\underline{\gamma}^{-i}$ (it may be an empty set or have several elements).

In our setting, it is easy to see that for each mobile i and each fixed strategy $\underline{\gamma}^{-i}$ for other players, the best response of mobile i is $\gamma_i = 0$ (unless $O(i) = \emptyset$ in which case, the best response is the whole interval $[0, 1]$). Thus the only possible equilibrium is that of $\gamma_i = 0$ for all i. To overcome this problem, we consider the following "punishing mechanism". in order to incite mobiles to cooperate.

Definition: *Consider a given set of policies $\underline{\gamma} = (\gamma, \gamma, \gamma, ...)$. If some mobile deviates and uses some $\gamma' < \gamma$, we define the punishing policy $\kappa(\gamma', \gamma)$ as the policy in which all mobiles decrease their forwarding probability to γ'.*

When this punishing mechanism is enforced, then the best strategy of a mobile i when all other mobiles use strategy γ is γ' that achieves

$$J(\gamma) := \max_{\gamma' \leq \gamma} U_i(\underline{\gamma}') \tag{2}$$

where $\underline{\gamma}' = (\gamma', \gamma', \gamma', ...)$.

Definition: *If some γ^* achieves the minimum in (2) we call the vector $\underline{\gamma}^* = (\gamma^*, \gamma^*, \gamma^*, ...)$ the equilibrium strategy (for the forwarding problem) under threats. $J(\gamma)$ is called the corresponding value.*

Remark: Note that $\gamma^* = 0$ is still a Nash equilibrium, a fact that will be used frequently in Section 5 where we obtain some structural properties of equilibrium strategy under threats.

3 Utilities for Symmetrical Topologies

By symmetrical topology we mean the case where f_i, g_i and ξ_i are independent of i. This implies that for any source-destination pair (s, d), there are two nodes

s' and d' such that the source-destination pairs (s', s) and (d, d') are identical to (s, d) in the sense that there view of the network is similar to that of (s, d). This implies that, under the punishment mechanism where all nodes have same forwarding probability, we have $p(s, d; \underline{\gamma}) = p(s', s; \underline{\gamma})$. Thus we can replace the rewards $f_i + g_i$ by another function that we denote $f(\cdot)$.

Consider $\underline{\gamma}$ where all entries are the same and equal to γ, except for that of mobile i. For a path $\pi(s, d)$ containing n intermediate nodes, we have $p(s, d; \underline{\gamma}) = \gamma^n$. Also, if a mobile i is $n + 1$ hops away from a source, $n = 1, 2, 3, ...$, and is on the path from this source to a destination (but is not itself the destination), then $p(s, d; i, \underline{\gamma}) = \gamma^n$. We call the source an “effective source” for forwarding to mobile i since it potentially has packets to be forwarded by mobile i. Let $h(n)$ be the rate at which all effective sources located $n + 1$ hops away from mobile i transmit packets that should use mobile i for forwarding (we assume that h is the same for all nodes). Let $\lambda^{(n)}$ denote the rate at which a source s creates packets to all destinations that are $n + 1$ hops away from it. Then we have

$$U_i(\underline{\gamma}) = \sum_{n=1}^{\infty} \lambda^{(n)} f(\gamma^n) - aE_f \sum_{n=1}^{\infty} h(n)\gamma^n. \tag{3}$$

The equilibrium strategy under threat is then the value of γ that maximizes the r.h.s.

Remark: If we denote by $\Lambda(z) = \sum_{n=1}^{\infty} z^n \lambda^{(n)}$ the generating function of $\lambda^{(n)}$ and $H(z) := \sum_{n=1}^{\infty} z^n h(n)$ the generating function of h. Then $\max_\gamma \Big(\Lambda(\gamma) - aE_f H(\gamma)\Big)$ is the value of the problem with threats in the case that f is the identity function.

4 Examples

In this section we present, by means of two examples, the effect of imposing the proposed punishment mechanism.

4.1 An Asymmetric Network

Consider the network shown in Figure 1. For this case nodes 1 and 4 have no traffic to forward. Note also that if we assume that $g_3(\cdot) \equiv 0$ in Equation 1 then node 3 has no incentive even to invoke the punishment mechanism for node 2. This will result in no cooperation in the network. Assume for the time being that $f_2(x) = g_3(x) = x$, i.e., f_2 and g_3 are identity functions. In this case it is seen that the utility functions for nodes 2 and 3 are, assuming $\lambda_{13} = \lambda_{24} = 1$, $U_2(\gamma_2, \gamma_3) = \gamma_3 - aE_f\gamma_2$ and $U_3(\gamma_2, \gamma_3) = \gamma_2 - aE_f\gamma_3$. When we impose the punishment mechanism, it turns out that the equilibrium strategy for the two nodes is to always cooperate, i.e., $\gamma_2 = \gamma_3$. This is to be compared with the TFT strategy of [3] which would imply $\gamma_2 = \gamma_3 = 0$.

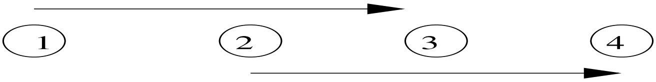


Fig. 1. An asymmetric network

4.2 A Symmetric Network

We consider here equally spaced mobile nodes on a circle and assume that each node i is a source of traffic to a node located L hops to the right, i.e. to the node $i+L$.

Let the rate of traffic generated from a source be λ. For this case, $h(n) = \lambda I_{\{n \leq L-1\}}$. Also, $\lambda^{(n)} = \lambda I_{\{n=L\}}$, for some λ. It follows from Equation 3 that the utility function for mobile i is $U_i(\underline{\gamma}) = \lambda f(\gamma^{L-1}) - aE_f \lambda \sum_{n=0}^{L-2} \gamma^n$. For $f(\cdot)$ an identity function, we see that $U_i(\underline{\gamma}) = \lambda \left[\gamma^{L-1} - aE_f(\gamma^{L-2} + \gamma^{L-3} + \ldots + \gamma + 1)\right]$. Note that if $L = 2$ and $a = \frac{1}{E_f}$, the utility function is independent of γ hence in this case the equilibrium strategy is any value of forwarding probability. Also, if $aE_f \geq 1$, the equilibrium strategy is $\gamma = 0$. We will have more to say on this in th next section where we study the structure of equilibrium strategy for symmetric network.

5 Structure of Equilibrium Strategy

In this section we undertake the study of dependence of the equilibrium strategy on the various system parameters. We restrict ourselves to the case of symmetric topologies. Symmetry of the problem along with the imposed punishment mechanism implies that the equilibrium strategy (the forwarding probabilities) will be same for all the nodes in the network. We denote this probability by γ^*.

This is to be understood as follows. When a node i computes its equilibrium strategy γ_i, it must consider the fact that the other nodes will respond with a punishing mechanism to its strategy. Thus, the problem faced by node i is *not* that of optimizing Equation 3 with respect to γ_i considering γ^{-i} fixed (which will lead to the trivial solution of $\gamma_i = 0$ as seen before). Owing to the punishment mechanism, node i should apriori assume that all the forwarding probabilities are same, i.e., $\gamma^{-i} = (\gamma_i, \ldots, \gamma_i)$. This makes the problem faced by node i a single variable optimization problem.

Though $f(\cdot)$ is concave in its argument ($p(\gamma)$, which is a polynomial in γ), $f(p(\gamma))$ may not be concave as a function of γ. For example, in the case of circular network above, $f(p(\gamma)) = p(\gamma) = \gamma^{L-1}$, convex in γ. Thus obtaining a direct structural result for γ^* seems to be hard for general $f(\cdot)$ and $p(\cdot)$. We can get some interesting insights using some approximations; this is the aim of present section. In particular, we study how γ^* depends on the *system paramaters*, L, $f(\cdot)$, $p(\cdot)$, a and E_f.

It is clear from the expression of the utility function that γ^* will depend on a and E_f only through their product. Let us introduce the notation $K := aE_f$.

It is also clear from the definition of utility function ($U_i(\gamma)$) that if either K or L is *large*, the equilibrium strategy of the game is at *smaller* γ. It is also intuitive that for *small* values of K (or L), a node may forward most of the requesting packets. In the following we characterize what value of K or L can be considered as *large* or *small.* Clearly this characterization will depend on $f(\cdot)$ and the network, i.e., $p(\cdot)$ and $H(\cdot)$.

5.1 Dependence of γ^* on K

The case of General Network and $f(\cdot)$. Consider the line starting at $\gamma = 0$ with a value $f(0)$ and having a slope $f'(0)p'(1)$ for $0 \leq \gamma \leq 1$. This slope is an upper bound on the slope of $f(p(\gamma)$ in the stability region since $p(\cdot)$ is convex and $f(\cdot)$ is concave. Thus the line constructed above is an upper bound to $f(p(\gamma))$ for $0 \leq \gamma \leq 1$. Consider also the line starting from $f(0)$ and having a slope $f'(1)p'(0)$ (this slope is clearly a lower bound on the slope of $f(p(\gamma))$. It is thus true that f lies in *between* these two lines. Thus we can get conditions under which the equilibrium strategy is less than (or equal to) the maximal possible value, i.e., the extreme point $\gamma = 1$.

Proposition 1. *If the network topology and $f(\cdot)$ satisfy $f'(0)p'(1) \leq Kh(1)$, then the equilibrium strategy is $\gamma^* = 0$.*

Proof: Follows from the construction of bounds above.

Remark: Above result shows that if K, i.e., the energy spent in reception and transmission is larger than a threshold (say $K^* = \frac{f'(0)p'(1)}{h(1)}$), it is best for the nodes to not forward packets at all *even under the punishing mechanism.* This is to be compared with the fact proved below that $\gamma = 0$ is *always* a *local* maximum for $L > 2$ for the circular network. Thus the above result gives a criteria when $\gamma = 0$ is also a *global* maximum.

The Case of the Circular Network with $f(p(\gamma)) = p(\gamma)$. For the case of circular network and $f(\cdot)$ being an identity function, we can say more about the dependence of γ^* on K. It is seen that for this particular case, $U_i(0) = -K$.

For the optimum, the derivative of $U_i(\gamma)$ with respect to γ should be zero, i.e.,

$$\frac{L-1}{K}\gamma^{L-2} = (L-1)\gamma^{L-3} + \ldots + 1.$$

So, if $K > L - 1$, the above requirement is not possible for $\gamma < 1$. Thus the only solution is either $\gamma = 0$ or $\gamma = 1$. But, as will be seen in later section, $\gamma = 0$ is always a local maximum. Thus, in absence of any other critical point (where $U_i'(\gamma) = 0$), the global maximum is at $\gamma^* = 0$. Thus, for $K > L - 1$, we have that $\gamma^* = 0$. This is, as argued before, reasonable, as for K *large*, a node will spend more energy to forward packets, hence will not do it for any packet it gets for forwarding; we thus obtain the expression for energy requirement that can be characterised as *large*, i.e., $K^* := L - 1$.

In getting the above bound on K we assumed that $f(x) = x$. The above reasoning is however true for any $f(\cdot)$ whenever $f'(0) \leq \frac{K}{L-1}$ for the circular networks, i.e., if $f'(\cdot) \geq \frac{L-1}{K}$ since $f(\cdot)$ is concave. Thus, for a fixed set of K and L, we get that $\gamma^* = 0$ for any $f(\cdot)$ that has slope $\leq \frac{K}{L-1}$. Looking at it from other point, we again see that if $K \geq \frac{L-1}{f'(0)}$ then $\gamma^* = 0$.

5.2 Dependence of γ^* on L

In the above we saw that in the case of a symmetric network and for a fixed L, if $K > K^*$, we get that $\gamma^* = 0$. The intuition for this result is that a node will have to spend more energy in forwarding the packets as compared to the reward it gets by its own packet forwarded by other nodes. However, if we fix K and increase the hop-length L, it is intuitive that γ^* will eventually start decreasing as a function of L. This is established in [9].

6 Algorithm for Computing the Equilibrium Strategy in a Distributed Manner

It is interesting to design distributed algorithms which can be used by the mobiles to compute the equilibrium strategy and simultaneously enforce the proposed punishment mechanism. The obvious desirable features of such an algorithm are that it should be decentralised, distributed scalability and should be able to adapt to changes in network.

We propose such an algorithm in this section. We present it, for ease of notation, for the case of symmetric network. Assume for the moment that $f(\cdot)$ is the identity function. In this case each node has to solve the equation (recall the notation of Section 3)

$$U'(\gamma) = \Lambda'(\gamma) - KH'(\gamma) = 0, \tag{4}$$

where the primes denote the derivatives with respect to γ. In general this equation will be nontrivial to solve directly. For the case of more general network, one needs to compute the derivative of the utility function of Equation 1, the rest of procedure that follows is similar.

Note that in the above expression we first assume that the forwarding probabilities of all the nodes in the network are same (say γ) and then compute the derivative with respect to this common γ. This is because in the node must take the effect of punishment mechanism into account while computing its own optimal forwarding probability, i.e., a node should assume that all the other nodes will use the same forwarding probability that it computes.

Thus, solving Equation 4 is reduced to a single variable optimization problem. Since the actual problem from which we get Equation 4 is a maximization problem, a node does a gradient *ascent* to compute its optimal forwarding probability. Thus, in its n^{th} computation, a node i uses the iteration

$$\gamma_i^{(n+1)} = \gamma_i^{(n)} + a(n)(\Lambda'(\gamma_i^{(n)}) - KH'(\gamma_i^{(n)})), \tag{5}$$

where $a(n)$ is a sequence of positive numbers satisfying the usual conditions imposed on the learning parameters in stochastic approximation algorithms [8], i.e.,

$$\sum_n a(n) = \infty \text{ and } \sum_n a(n)^2 < \infty.$$

The relation to stochastic approximation algorithm here is seen as follows: the network topology can be randomly changing with time owing to node failures/mobility et cetera. Thus a node needs to appropriately modify the functions $\Lambda(\cdot)$ and $H(\cdot)$ based on its most recent view of the network (this dependence of $\Lambda(\cdot)$ and $H(\cdot)$ on n is suppressed in the above expression).

It is known by the o.d.e. approach to stochastic approximation algorithm that the above algorithm will asymptotically track the o.d.e. [8]:

$$\dot{\gamma}_i(t) = \Lambda'(\gamma_i(t)) - KH'(\gamma_i(t)), \tag{6}$$

and will converge to one of the *stable* critical points of o.d.e. of Equation 6. It is easily seen that a local maximum of the utility function forms a stable critical point of Equation 6 while any local minimum forms an unstable critical point. Thus the above algorithm inherently makes the system converge to a local maximum and avoids a local minimum.

However, it is possible that different nodes settle to different local maxima (we have already seen that there can be multiple maxima). The imposed punishment mechanism then ensures that all the nodes settle to the one which corresponds to the lowest values of γ. This is a desirable feature of the algorithm that it inherently avoids multiple simultaneous operating points. An implementation of the punishment mechanism is described next.

6.1 Distributed Implementation of the Punishment Mechanism

An implementation of punishment mechanism proposed in Section 2 requires, in general, a node to know about the misbehaving node in the network, if any. Here we propose a simple implementation of the punishment mechanism which requires only local information for its implementation.

Let $\mathcal{N}(i)$ be the set of neighbours of node i. Every node computes its forwarding policy in a distrubuted manner using the above mentioned stochastic approximation algorithm. However, as soon as a neighboring node is detected to misbehave by a node, the node computes its forwarding policy as follows:

$$\gamma_i^* = \min\{\gamma_i, \min_{j \in \mathcal{N}(i)} \hat{\gamma}_j\} \tag{7}$$

where γ_i and $\hat{\gamma}_j$ represents, respectively, the forwarding policy adopted by node i and the estimate of node j's forwarding probability available to node i. γ_i^* represents the new policy selected by node i. Note here that γ_i is still computed using iteration of Equation 5. We are also assuming here that a node can differentiate between a misbehaving neighbouring node and the failure/mobility of a neighbouring node.

This punishment propagates in the network until all the nodes in the network settle to the common forwarding probability (corresponding to that of the misbehaving node). In particular, the effect of this punishment will be seen by the misbehaving ndoe as a degradation in its own utility. Suppose now that the misbehaving node, say n_i, decides to change to a cooperative behavior: at that point, it will detect and punish its neighbors because of the propagation of the punishment that induced its neighbouring nodes to decrease their forwarding policy. Thus, the intial punishment introduces a negative loop and the forwarding policy of every node of the network collapses to the forwarding policy selected by the misbehaving node. Since now every node in the network has same value of forwarding probability, none of the nodes will be able to increase its forwarding probability even if none of the node is misbehaving now.

An example of this phenomenon can be seen from the network of Figure 1. Assume that $\gamma_2 = \gamma_3 = \gamma$ and now node 2 reduces γ_2 to a smaller value γ'. Owing to the punishment mechanism, node 3 will respond with $\gamma_3 = \gamma'$. This will result in a reduced utility for node 2 which would then like to increase γ_2. But, since $\gamma_3 = \gamma'$, the punishing mechanism would imply that $\gamma_2 = \gamma'$ as well. This *lock-in* problem is avoided by the solution proposed below.

We modify our algorithm using timers to account for the above mentioned effect. Details are given in [9].

7 Numerical Results

In this section we present numerical results from a MATLAB implementation of the proposed algorithm. We consider a circular network with N mobiles where each mobile is a source of a traffic stream having as its destination the mobile that is L hops away. The numerical results are meant to validate the structural results obtained in the paper and also to indicate the possibility of practical implementation of the proposed punishment scheme. The particular choice of function $f(\cdot)$ are motivated by need to facilitate a better visual presentation. The plots of this section show the value of γ^* computed in the n^{th} iteration of the algorithm vs. the iteration number. The sequence of learning parameters is chosen to be $a(n) = \frac{1}{n \ln(n)}$ (we used various other options also, the results were similar). In the simulations we assume that each node has a lower bound $\gamma_{min} > 0$ on the value of forwarding probability. We take $\gamma_{min} = 0.01$ in results reported here.

7.1 Structural Results

We now validate against simulations the structural results obtained in the paper. Figure 2 depicts the distributed computation of γ^* using the proposed algorithm. We use $f(x) = \ln(100x + 1)$ and $L = 3$. The various curves are obtained by varying the value of K. We see that γ^* is indeed a dereasing function of K. Similar observation is made from Figure 3 where we fix $K = 0.2$ and $f(x) = (x + 0.0005)^{0.2}$ and vary L. It is seen that γ^* decreases with increasing L and for large value of $L (= 11)$, $\gamma^* = \gamma_{min}$.

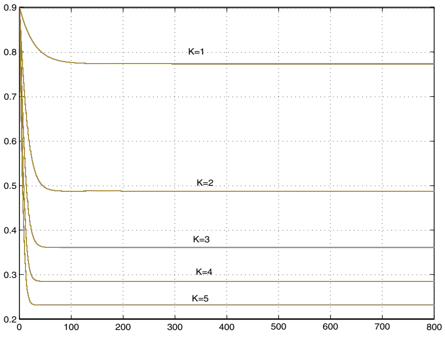


Fig. 2. $L = 3$ and K is varied. The function $f(x) = \ln(100x + 1)$

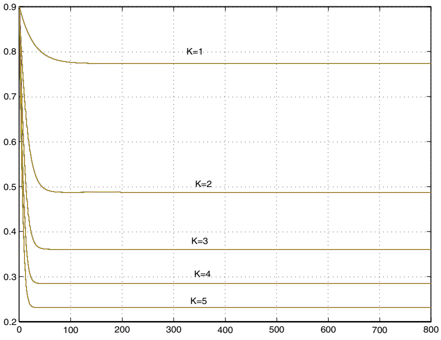


Fig. 3. $K = 0.2$ and L is varied. The function $f(x) = (x + 0.0005)^{0.2}$

7.2 Results with Punishment Mechanism Invoked

Figure 4 gives the evolution of the estimated of γ^* of various nodes for the circular network with $N = 6$, $L = 3$, $K = 2$ and $f(x) = \ln(100x + 1)$. Node 1 starts misbehaving at iteration number 100 and continues to misbehave till iteration number 130; during this period node 1 keeps a forwarding probability of 0.1. Note that during this period all the other nodes decrease their individual forwarding probability to the value used by node 1, i.e., 0.1. The jumps in the forwarding probabilities during these period are because of the implementation of timer for punishment in the individual nodes (detailed in previous section). The timer value was set to 10 simulation slots. Soon after node 1 stops misbehaving, the forwarding probabilities of all the other nodes increase to settle to the optimal value 0.48. Note here that the convergence of the gradient algorithm is fast enough and that the nodes restart the learning sequence $a(n)$ after each timer expiry. Also shown in the figure is that node 2 misbehaves in the period between iteration 600 and 650.

7.3 The Asymmetric Network of Figure 1

We now consider the case of asymmetric network of Figure 1. We assume that $f_2(x) = g_3(x) = \sqrt(x)$ and that $g_3(\cdot) \equiv 0$. In this case it is seen that the utility functions for nodes 2 and 3 are, assuming $\lambda_{13} = \lambda_{24} = 1$, $U_2(\gamma_2, \gamma_3) = \sqrt(\gamma_3) - aE_f\gamma_2$ and $U_3(\gamma_2, \gamma_3) = \sqrt(\gamma_2) - aE_f\gamma_3$. On imposing the ponushment mechanism for this case it is seen that, the optimal value of forwarding probabilities of node 2 and node 3 are $\gamma_2 = \gamma_3 = \frac{1}{\sqrt{2aE_f}}$. Figure 5 gives the evolution of the estimates of γ_2 and γ_3 for this network for different values of $K = aE_f$. The minimum value of γ_i was taken to be 0.1 and the maximum value was 0.9. Note from the figure that for $K = 1$, the equilibrium strategy of $\frac{K}{2}$ as obtained above is achieved. For large (resp. small) value of K, the equilibrium strategy is at γ_{max} (resp. γ_{min}).

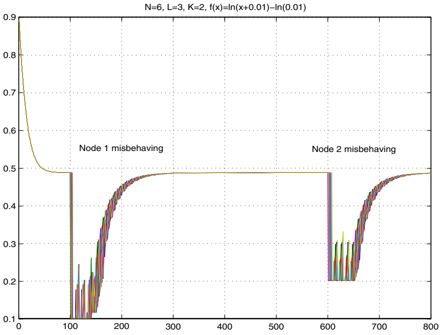


Fig. 4. $N = 6$, $K = 2$, $L = 3$, $f(x) = \ln(100x + 1)$. Nodes 1, 2 misbehave in nonoverlapping intervals

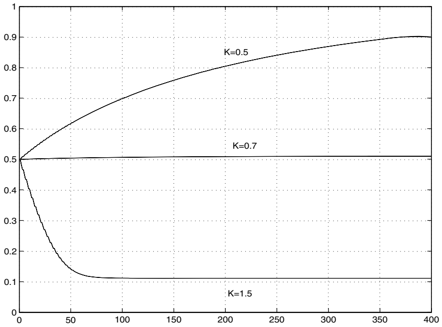


Fig. 5. Evolution of the estimates of γ_2 and γ_3 for the network of Figure 1

8 Conclusion

We use the framework of non-cooperative game theory to provide incentives for collaboration in the case of wireless Ad-hoc networks. The incentive proposed in the paper is based on a simple punishment mechanism that can be implemented in a completely distributed manner with very small computational complexity. The advantage of the proposed strategy is that it results in a less "aggressive" equilibrium in the sense that it does not result in a degenerate scenario where a node either forwards all the requested traffic or does not forward any of the request.

Some structural results relating the equlibrium strategy to the system parameters were also presented and were verified using an implementation of the punishing mechanism.

References

1. D. Dutta, A. Goel and J. Heidemann, "Oblivious AQM and Nash Equilibria", IEEE Infocom, 2003.
2. J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling incentives for collaboration in mobile Ad-hoc networks. In *Proceedings of WiOpt'03*, Sophia-Antipolis, France, 3-5, March 2003.
3. M. Félegyházi, L. Buttyán and J. P. Hubaux, "Equilibrium analysis of packet forwarding strategies in wireless Ad-hoc entworks – the static case", PWC 2003 Personal Wireless Communications, Sept. 2003, Venice, Italy.
4. P. Michiardi and R. Molva. A game theoretical approach to evaluate cooperation enforcement mechanisms in mobile Ad-hoc networks. In *Proceedings of WiOpt'03*, Sophia-Antipolis, France, 3-5, March 2003.

5. L. Samuelson, "Subgame Perfection: An Introduction," in John Creedy, Jeff Borland and Jrgen Eichberger, eds., Recent Developments in Game Theory, Edgar Elgar Publishing, 1992, 1-42.
6. V. Srinivasan, P. Nuggehalli, C. F. Chiasserini and R. R. Rao, "Cooperation in wireless Ad-hoc networks", *Proceedings of IEEE Infocom*, 2003.
7. A. Urpi, M. Bonuccelli, and S. Giordano. Modelinig cooperation in mobile Ad-hoc networks: a formal description of selfishness. In *Proceedings of WiOpt'03*, Sophia-Antipolis, France, 3-5, March 2003.
8. H. J. Kushner and G. Yin, "Stochastic Approximation Algorithms and Applications," Springer-Verlag, 1997.
9. E. Altman, P. Michiardi and R. Molva. Non-cooperative Forwarding in Ad-hoc Networks. INRIA Report No. RR-5116, Sophia-Antipolis, France, February 2004.

Efficient Broadcasting in Ad Hoc Networks Using Directional Antennas[*]

Fei Dai and Jie Wu

Department of Computer Science and Engineering,
Florida Atlantic University,
Boca Raton, FL 33431
jie@cse.fau.edu

Abstract. Using directional antennas to conserve bandwidth and energy consumption in ad hoc networks is becoming popular in recent years. However, applications of directional antennas for broadcasting have been limited. We propose a novel broadcast protocol called *directional self-pruning* (DSP) for ad hoc networks using directional antennas. DSP is a non-trivial generalization of an existing localized deterministic broadcast protocol using omnidirectional antennas. Compared with its omnidirectional predecessor, DSP uses about the same number of *forward nodes* to relay the broadcast packet, while the number of *forward directions* that each forward node uses in transmission is significantly reduced. With the lower broadcast redundancy, DSP is more bandwidth- and energy-efficient. DSP is based on 2-hop neighborhood information and does not rely on location or angle-of-arrival (AoA) information. DSP is a pure localized protocol. We prove that the expected number of forward nodes in DSP is within a constant factor of the minimal value in an optimal solution. Our simulation results show that DSP can reduce the transmission cost by 30%–65%.

1 Introduction

Using directional antennas to conserve bandwidth and energy consumption in wireless communications is becoming popular in recent years [1]. Compared with the omnidirectional antennas, a smart antenna can form directional *beams* for both transmission and reception, which achieves better signal-to-noise ratio (SNR) and reduces interference. Many network protocols have been proposed for using directional antennas in ad hoc networks [2, 3, 4, 5, 6]. However, most of them focused on the MAC layer, and research on the application of directional antennas in unicasting and broadcasting has been limited.

Broadcasting is frequently used in ad hoc networks for data dissemination and on-demand route discovery. Blind flooding has high cost and excessive redundancy, which causes the broadcast storm problem [7]. Both probabilistic approaches [7] and deterministic approaches [8, 9, 10, 11, 12] have been proposed for efficient broadcasting in

[*] This work was supported in part by NSF grants CCR 0329741, CNS 0422762, CNS 0434533, ANI 0073736, and EIA 0130806.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 499–510, 2005.

ad hoc networks. Probabilistic approaches need relatively high broadcast redundancy to maintain an acceptable delivery ratio. Deterministic approaches select a few *forward nodes* based on neighborhood information to achieve full delivery. Most deterministic broadcast schemes in ad hoc networks are *localized.* A localized algorithm determines the status of each node (forward or non-forward) based on its k-hop neighborhood information, where k is a small constant.

Although deterministic algorithms are more efficient than probabilistic approaches, their broadcast redundancy is not minimized. Wireless nodes with directional antenna can control their radiation pattern to reduce broadcast redundancy. Several protocols [13, 14, 15, 16] have been proposed for efficient broadcasting using directional antennas. However, most of them are probabilistic approaches, depend on location or AoA information, and assume specific antenna models. All those protocols assume the omnidirectional reception mode. In this paper, we propose a novel broadcast protocol called *directional self-pruning* (DSP), which extends an omnidirectional broadcast protocol (called self-pruning) [17]. Extending the omnidirectional self-pruning scheme to use directional antennas is non-trivial. We show that the original self-pruning algorithm in [17] must be enhanced carefully to avoid broadcast failure without be overly conservative. Compared with its omnidirectional predecessor, DSP minimizes the interference and energy consumption by switching off transmission in unnecessary directions. Our simulation results show that DSP can reduce the transmission cost by 30%–65%.

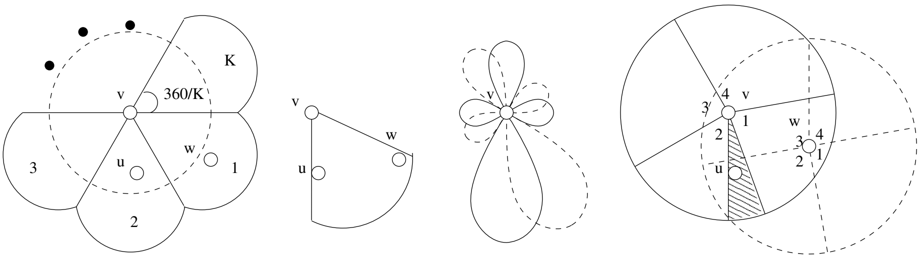
In DSP, each node is equipped with only 2-hop neighborhood information (or simply 2-hop information), which is collected via two rounds of "Hello" exchanges among neighbors. The direction information (i.e., how to form a directional beam to reach a specific neighbor) is included in the 2-hop information and does not cause extra overhead to collect. DSP uses a general antenna model with fewer assumptions than existing models. The main contributions of this paper are as follows:

1. Combine directional antennas with the latest broadcast techniques to minimize the broadcast redundancy in ad hoc networks.
2. Provide a general antenna model and a directional neighbor discovery scheme that does not rely on any location or AoA information.
3. Conduct both theoretical and simulation study to evaluate the performance of DSP.

The remainder of this paper is organized as follows. Section 2 reviews existing broadcast schemes. In Section 3, we introduce a general antenna model, a neighborhood discovery scheme, and a formal definition of the problem. Section 4 discusses the DSP algorithm and properties. Simulation results are presented in Section 5. Section 6 concludes this paper.

2 Related Work

Many deterministic broadcast schemes have been proposed for ad hoc networks using omnidirectional antennas. A deterministic broadcast algorithm is equivalent to an algorithm that forms a *connected dominating set* (CDS). The problem of finding a minimal CDS was proved NP-complete. Approximation algorithms exist, but are either centralized [18], cluster-based [19], or location-based [20]. Centralized and cluster-based



(a) ideally sectorized (b) adjustable cone (c) irregular beam pattern (d) a general model

Fig. 1. Directional antenna models

algorithms have slow convergency in mobile networks. Location-based algorithms rely on external devices such as GPS receivers, which cause extra cost. Localized broadcast algorithms can be further divided into *neighbor designating* algorithms and *self-pruning* algorithms. In neighbor-designating [8, 9, 11], each forward node selects a few 1-hop neighbors as new forward nodes to cover its 2-hop neighbors. In self-pruning [10, 12, 21], each node determines it own status (forward or non-forward). A generic self-pruning scheme was proposed in [17].

Application of directional antennas in localized broadcasting is limited in literature. Most of them are probabilistic approaches [14, 15, 16]. Choudhury and Vaidya [14] proposed to reduce the broadcast redundancy in relaying routing request by switching off transmissions in directions toward the last forward node. Hu, Hong, and Hou [15] presented three schemes to improve the broadcast efficiency. In the first scheme, each node switches off its transmission beams towards known forward nodes. In the second and third schemes, each forward node designates only one neighbor as a forward node in each direction. In the third scheme, the selection of forward nodes is aided by location information. Shen et al [16] devised directional versions of probabilistic protocols in [7]. Only two localized deterministic schemes were proposed [13, 16]. Lim and Kim's neighbor elimination [8] was extended in [16], where each node switches off transmission in a direction, if all neighbors in this direction are also neighbors of a known forward node. In [13], each node forms a single beam with an adjustable width to reach all neighbors that are not covered by transmissions of known forward nodes. Location information is used to calculate the angle and orientation of the transmission beam.

3 Preliminaries

3.1 Antenna Model

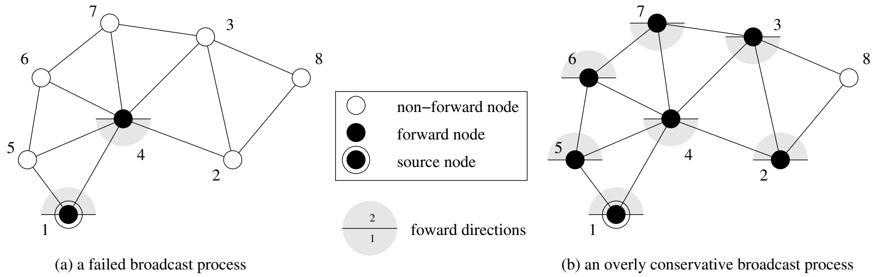
Two beam-forming techniques exist: *switched beam* and *steerable beam* [1]. Switched beam systems use fixed antenna patterns to transmit to or receive from specific directions. A popular antenna model for those systems is ideally sectorized [14, 15, 16], as shown in Figure 1 (a). The neighborhood of each node v is equally divided into K non-overlapping sectors. Each node can switch on one or several sectors for transmission or reception. Aligned sectors are assumed in most existing protocols. Steerable beam systems can adjust the bearing and width of a beam to transmit to or receive from certain neighbors. The corresponding antenna mode is a adjustable cone [13], as shown in Figure 1 (b). Most protocols also assume an omnidirectional mode, but the transmission range in omnidirectional mode (represented by the dashed circle in Figure 1 (a)) is substantially smaller than that in directional mode. Both antenna models assume regular beam shapes. In practical systems, antenna patterns have irregular shapes due to the existence of side lobes (as shown in Figure 1 (c)).

This paper uses a general antenna model that does not rely on a specific beam-forming technique. As shown in Figure 1 (d), each node can transmit and receive in K *directions* with id's $1, 2, \ldots, K$. The shape of each direction can be irregular, overlapping (see the shadowed area), and unaligned. Each node can transmit in one direction or several directions via sweeping [14]. The reception mode can be omnidirectional (default) or directional. For each node v, $N_i(v)$ denotes v's neighbor set direction i, and $N(v) = N_1(v) \cup N_2(v) \cup \ldots \cup N_K(v)$ is v's complete neighbor set. A neighbor may appear in several directions when there is an overlapped area. For each neighbor u, its directions with respect to node v is $D_{v \to u} = \{i | u \in N_i(v)\}$. For example, in Figure 1 (d), $N(v) = \{u, w\}$, where $u, w \in N_1(v)$ and $u \in N_2(v)$. Therefore, $D_{v \to u} = \{1, 2\}$ and $D_{v \to w} = \{1\}$. The network is viewed as a graph $G = (V, E)$, where V is the set of nodes, and E is the set of bidirectional links. A wireless link $(u, v) \in E$ if and only if $v \in N(u)$ and $u \in N(v)$. We assume the network is symmetric and connected via bidirectional links.

3.2 Directional Neighborhood Discovery

In directional neighborhood discovery, each node sends periodical "Hello" messages to its neighbors. Each "Hello" message is transmitted in all directions. By collecting "Hello" messages from its neighbors, each node v can assemble its 1-hop information, including id's and directions of its neighbors. Note that the direction for v to reach a neighbor u is still unknown at that time. The 1-hop information is exchanged among neighbors in the next round of "Hello" messages. By assembling the 1-hop information of v its neighbors, node v can construct its 2-hop (direction) information, which is a subgraph of G derived from v's closed neighbor set $N[v] = N(v) \cup \{v\}$, and direction $D_{u \to w}$ for any two nodes $u, v \in N[v]$. Note that v's 2-hop neighbors are excluded from the 2-hop information, because the direction from a 1-hop neighbor to any 2-hop neighbor is unknown.

In the above scheme, each "Hello" message is sent out K times in K directions at each node. However, given the same neighborhood area, the cost of each directional transmission is roughly $1/K$ that of an omnidirectional transmission. The total cost of the directional neighborhood discovery is similar to that of the traditional scheme using omnidirectional "Hello" messages. This scheme also works when there are obstacles, as the neighbor and direction information is retrieved from real signal reception instead of being computed from an ideal antenna pattern. We assume that node movement, in terms of changing positions or turning on their axes, is relatively slow with respect to the "Hello" interval, so that 2-hop information collected at each node is up-to-date. We also assume that packet collision is avoided via an ideal MAC layer. For clarity, we use ideally sectorized direction shapes in examples. Nevertheless, all results in this paper work for the general antenna model as shown in Figure 1 (d).



(a) a failed broadcast process

(b) an overly conservative broadcast process

Fig. 2. Problems in converting OSP to DSP

3.3 Efficient Broadcasting

For each broadcasting, a few *forward nodes* are selected to forward in some *forward directions*. We define the *forward scheme*, F, as a function on V, where $F(v)$ is the set of v's forward directions. Given F, we say a destination d is *reachable* from a source s, if $s = d$ or there exists a *forward path* $P : (v_1 = s, v_2, \ldots, v_l = d)$ satisfying that every node in P forwards to the direction of its successor. A forward scheme F achieves *full delivery* if all nodes in the network are reachable from s. Given an antenna model, we define the *transmission cost* of a forward scheme as $|F| = \sum_{v \in V} |F(v)|$.

Efficient Broadcasting. *Given a number of antenna directions K, network G, and source s, find the forward scheme F that achieves full delivery with minimum transmission cost $|F|$.*

Efficient broadcasting using omnidirectional antennas is a special case of the above problem with $K = 1$, which is known to be NP-complete. The efficient broadcasting problem with a particular $K \geq 2$ in a geometric graph is conjectured to be NP-complete. Our objective is to find a localized solution with a low average transmission cost.

We first review the *omnidirectional self-pruning* (OSP) as a trivial solution to the above problem. In OSP, each node computers the coverage of its neighborhood after receiving the packet from one or several *known forward nodes*. In node v's local view, a node w is *covered* if: (1) w is a known forward node, (2) w is a neighbor of a known forward node, or (3) w is a neighbor of a covered node with a higher id than v. If v has uncovered neighbors, it becomes a forward node and transmits in all directions; otherwise, it does nothing. It was proved in [17] that OSP guarantees full delivery.

4 Directional Self-pruning

Intuitively, a forward node only needs to transmit in directions of uncovered neighbors to conserve transmission cost. However, this "optimization" is over aggressive and causes broadcast failures. As shown in Figure 2 (a), if node 4 forwards in direction 1 only, neither node 5 nor 2 will forward the packet. Nodes 3, 6, 7, and 8 will never received the packet.

A solution to the above problem is for each forward node to piggyback its forward directions to the data packet. In computing coverage, a neighbor w of a forward node

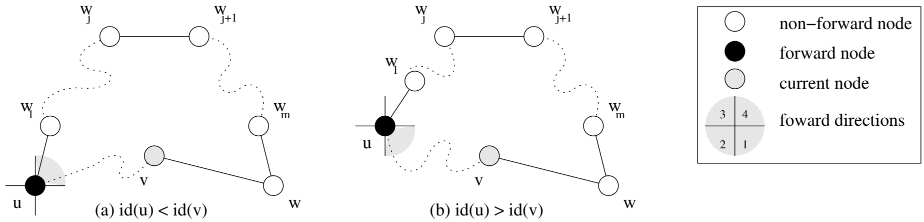


Fig. 3. Replacement paths

u is not covered unless it is within a forward direction of u. This rule ensures full delivery, but is overly conservative and causes unnecessary transmissions. As shown in Figure 2 (b), nodes 2, 3, 5, 6, and 7 become forward nodes under the new rule. However, transmissions of nodes 2 and 3 are redundant.

Directional self-pruning (DSP) uses a refined coverage rule to achieve both full delivery and high efficiency. A neighbor w of node v is viewed covered if and only if w:

1. is a known forward node,
2. is a neighbor of a known forward node u that has transmitted in w's direction, or
3. is a neighbor of a covered node with a higher id than v.

As shown in Figure 3, a covered neighbor is either a forward node or connected to a forward node via a *replacement path* $(u, w_1, w_2, \ldots, w_m, w)$, where $id(w_i) > id(v)$. Two scenarios exit: (a) $id(u) < id(v)$, then w_1 must be within a forward direction of u in order to apply term 2; (b) $id(u) > id(v)$, then w_1 can be out of u's forward directions by applying term 3.

Based on the new rule, node 5 in Figure 2 can no longer view node 6 as covered, because node 4 has not transmitted in direction 2 and, in addition, node 4 has a lower id than node 5. Therefore, node 5 becomes a forward node. Similarly, nodes 6, 7, and 3 become forward nodes and ensure full delivery. On the other hand, node 2 can still view nodes 3 and 8 as covered, because both nodes are connected via a replacement path to node 4, which has a higher id than node 2. Therefore, node 2 becomes a non-forward node.

Algorithm 1. Directional Self Pruning (at each node v)

```
Compute the set $U$ of uncovered nodes.
If $U = \emptyset$, then $v$ becomes a non-forward node.
Otherwise, $v$ becomes a forward node. Its set of forward directions is $\{d_{v \to w} | w \in U\}$.
```

Algorithm 1 gives the DSP algorithm. For each uncovered neighbor w, at least one direction $d_{v \to w} \in D_{v \to w}$ must become a forward direction in $F(v)$. If directions are overlapping, each uncovered node may be within several directions (i.e., $|D_{v \to w}| > 1$). In this case, a greedy heuristic algorithm for the set coverage problem [18] can be used to select a minimum $F(v)$ that covers all nodes in $N(v) - C$.

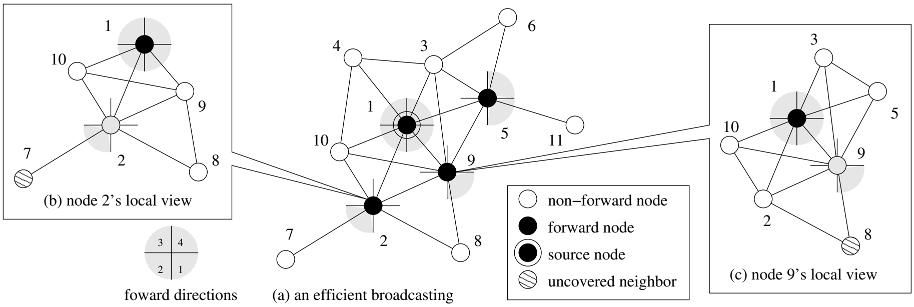


Fig. 4. Directional self-pruning

Figure 4 illustrates DSP in a network with 11 nodes, each node has ideally sectorized directions with $K = 4$. The source node 2 transmits in all four directions. Node 2 transmits in only one direction, because in its local view (as shown in Figure 4 (b)), all neighbors except node 7 are covered. Meanwhile, there is an uncovered node 8 in node 9's local view (as shown in Figure 4 (c)). Therefore, node 9 becomes a forward node and transmits in direction 1. Similarly, node 5 has two uncovered nodes 6 and 10, and transmits in directions 3 and 4. Each node receives the broadcast packet exactly once, except for the source node.

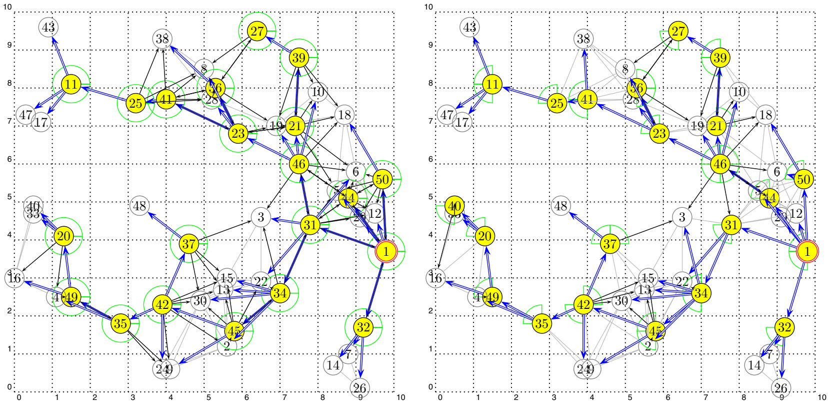
Theorem 1. *The forward scheme determined by DSP achieves full delivery.*

Proof. By contradiction, suppose there is at least one node that is unreachable from the source s. Let U be the set of "border" nodes that (1) is reachable from the source node, and (2) has an unreachable neighbor. U is not empty in a connected network. Let v be the node with the highest id in U, and w an unreachable neighbor of v. Since v has not transmitted in w's direction, node w must be covered in v's local view. However, we show that w cannot be covered, as none of the three terms in the refined coverage rule applies:

1. *w is a known forward node*, which implies that w is reachable from s.
2. *w is a neighbor of a known forward node u that has transmitted in w's direction.* In this case w is reachable from s via a forward path through u.
3. *w is a neighbor of a covered node with a higher id than v.* There are only two possible scenarios, as shown in Figure 3. In both cases, the unreachable node w is connected to a reachable node w_1 via a path $P : (w_1, w_2, \ldots, w_m, w)$, where each w_i $(1 \leq i \leq m)$ has a higher id than v. There is at least one node w_j in P that has an unreachable neighbor w_{j+1} (here we view w as w_{m+1}). That is, $w_j \in U$, which contradicts the assumption that v has the highest id in U.

Theorem 2. *In random ad hoc networks, the expected number of forward nodes in DSP is $O(1)$ times that in an optimal solution.*

Proof of Theorem 2 is omitted due to the limit of space. Whether a bound exists on the number of forward directions remains an open problem.



(a) OSP

(b) DSP ($K = 4$)

Fig. 5. Sample broadcast processes from source node 1. Gray nodes are forward nodes and white nodes are non-forward nodes. Pies surrounding forward nodes represent forward directions. Arrows represent receptions of the broadcast packet, where double lines are first receptions, and single lines redundant receptions

5 Simulation

5.1 Simulation Environment

We simulated DSP, OSP, and blind flooding on a customized simulator. Simulations are conducted in random networks with 20–200 nodes deployed in a 100×100 area. We use two fixed ranges $r = 25$ and $r = 50$, which correspond to relatively sparse and dense networks, respectively. Only connected networks are used in the simulation; disconnected networks are discarded. All nodes have an ideally sectorized antenna pattern with K sectors ($2 \leq K \leq 16$). We assume no mobility or collision. The following measures are compared: (1) *efficiency* in terms of the number of forward nodes and normalized transmission cost $|F|/K$, (2) *redundancy ratio*, i.e., average number of receptions per node, and (3) average *routing distance* in hops. The 90% confidence intervals of these measures are within $\pm 1\%$.

Figure 5 illustrates executions of omnidirectional and directional self-pruning algorithms in a random network with 50 nodes and an average node degree of 6. OSP (shown in Figure 5 (a)) uses 21 forward nodes. Its normalized transmission cost is also 21. Its redundant ratio is 2.74. The average routing distances is 3.70. DSP (shown in Figure 5 (b)) uses 22 forward nodes and 34 forward directions, which corresponds to a normalized transmission cost of 8.5 with 4 directions. Its redundant ratio is 1.56. The average routing distances is 3.72. In this example, DSP has a similar number of forward nodes and routing distance to OSP, and has lower normalized transmission cost and redundant ratio.

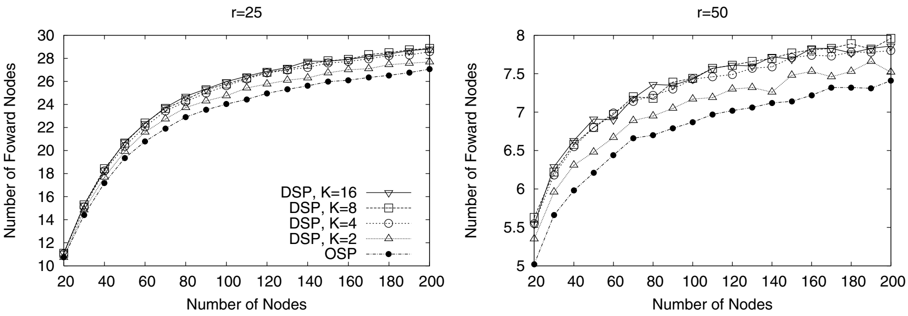


Fig. 6. Number of forward nodes

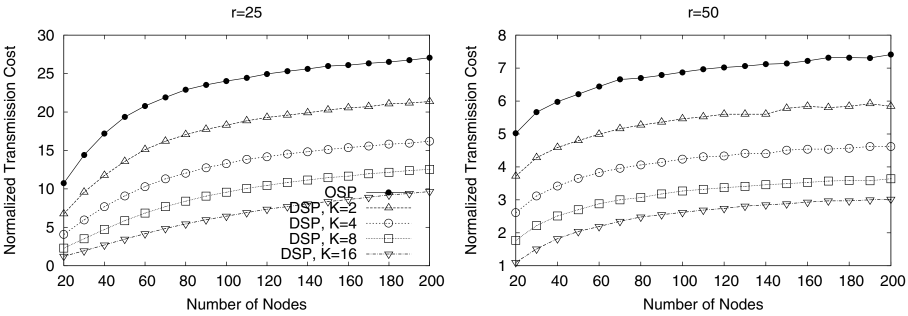


Fig. 7. Normalized transmission cost

5.2 Simulation Results

Efficiency. Figures 6 and 7 compare the broadcast efficiency of OSP and DSP. The left graph in Figure 6 shows the number of forward nodes in relatively sparse networks ($r = 25$), and the right graph shows results in relatively dense network ($r = 50$). The same layout is used in the following figures. For all network types, DSP uses more forward nodes than OSP. DSP uses about 5% more forward nodes than OSP when $K = 2$, and about 10% more forward nodes when $K = 4, 8$, or 16. It is because fewer nodes can be marked as covered in each node's local view in DSP. It is because fewer nodes can be marked as covered in each node's local view in DSP. Based on the refined coverage rule, a neighbor w of a known forward node u in node v's local view may not be covered if w is in a non-forward direction of u, and u has a lower id than v. Using more directions may produce more uncovered nodes.

Figure 7 shows the transmission cost of different schemes. For all network types, the normalized transmission cost of DSP with $K = 2, 4, 8$, and 16 is about 70%, 55%, 45%, 35% that of OSP. The fraction of non-forward directions increases as more directions are used to create finer divisions. On the other hand, the gain in broadcast efficiency is not a linear function of K. Considering the complexity of forming many beam patterns, using $K = 4$ or $K = 8$ is good enough to conserve bandwidth and energy consumption.

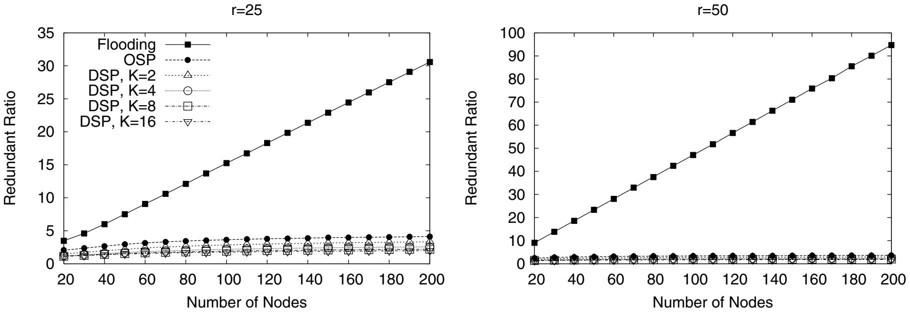


Fig. 8. Redundant ratio

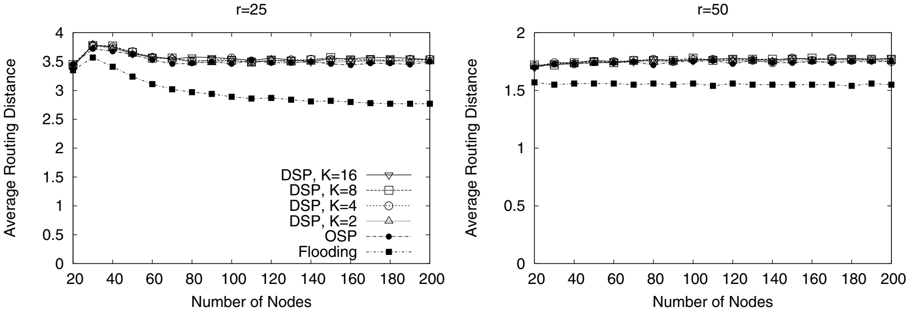


Fig. 9. Average routing distance

Redundancy. Figure 8 shows the redundancy of blind flooding, OSP, and DSP. While the redundancy ratio of blind flooding increases as the number of nodes increases, redundant ratios of the self-pruning schemes remain low. Specifically, the redundant ratio of OSP is about 4, and that of DSP with $2 \leq K \leq 16$ is between 1.8 to 3.5. The redundancy ratio of DSP is smaller than that of OSP, but the difference is not as large as in the case of normalized transmission cost. It is because some forward nodes have "empty" directions. As DSP has very low redundancy with a larger K, it is very efficient in conserving the bandwidth resource. On the other hand, it may suffer a reliability problem in a real environment with packet losses caused by mobility, collision, and signal fading. In such a case, either a small K or some reliability mechanisms, such as acknowledgement, should be used.

Routing distance. Figure 9 compares average routing distances. The difference between DSP and OSP is very small. Self-pruning algorithms have larger average routing distances than blind flooding. In all types of networks, the average routing distance of blind flooding is about 20% shorter than those of self-pruning algorithms. That is, if a self-pruning algorithm, omnidirectional or directional, is used to disseminate route request packets in a reactive routing protocol, the discovered route is expected to be 20% longer than the one discovered via blind flooding. In this case, tradeoffs must be done to balance the route discovery cost and the cost of transmitting data packets along a longer route. However, once self-pruning is selected to disseminate route request packets, using DSP will not further increase the length of the discovered route.

Simulation results can be summarized as follows: (1) DSP uses slightly more forward nodes than OSP, but has a much lower bandwidth and energy consumption. (2) The redundant ratio of DSP is 50%–89% that of OSP. (3) The average routing distance of DSP is very close to that of OSP, and is about 20% longer than the optimal distance.

6 Conclusion

We have proposed an efficient broadcast protocol for ad hoc networks using directional antennas. This protocol, called directional self-pruning (DSP), is a non-trivial generalization of an existing localized deterministic broadcast protocol using omnidirectional antennas. Compared with its omnidirectional predecessor, DSP achieves much lower broadcast redundancy and conserves bandwidth and energy consumption. DSP is based on 2-hop topology information and does not rely on any location or angle-of-arrival (AoA) information. We proved that the average number of forward nodes in DSP is within a constant factor of the minimal value in an optimal solution.

In future work, we plan to expand the proposed scheme to support neighbor designating protocols such as MPR and its variations [8,9,11]. Another task is the probabilistic analysis on the number of forward directions in random ad hoc networks. We expect that the average number of forward directions in DSP is also bounded as the number of forward nodes, but the technique used to calculate the number of forward nodes does not apply to the case of forward directions. We also plan to simulate DSP in more realistic networks with mobility and collisions, and embed DSP in an on-demand routing protocol to evaluate the overall routing performance.

References

1. Ramanathan, R.: On the performance of ad hoc networks with beamforming antennas. In: Proc. of ACM MobiHoc. (2001) 95–105
2. Bao, L., Garcia-Luna-Aceves, J.J.: Transmission scheduling in ad hoc networks with directional antennas. In: Proc. of ACM MobiCom. (2002) 48–58
3. Choudhury, R.R., Yang, X., Ramanathan, R., Vaidya, N.H.: Using directional antennas for medium access control in ad hoc networks. In: Proc. of ACM MobiCom. (2002) 59–70
4. Korakis, T., Jakllari, G., Tassiulas, L.: A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks. In: Proc. of ACM MobiHoc. (2003) 98–107
5. Roy, S., Saha, D., Bandyopadhyay, S., Ueda, T., Tanaka, S.: A network-aware MAC and routing protocol for effective load balancing in ad hoc wireless networks with directional antennas. In: Proc. of ACM MobiHoc. (2003)
6. Takai, M., Martin, J., Ren, A., Bagrodia, R.: Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In: Proc. of ACM MobiHoc. (2002) 183–193
7. Tseng, Y.C., Ni, S.Y., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. Wireless Networks **8** (2002) 153–167

8. Lim, H., Kim, C.: Multicast tree construction and flooding in wireless ad hoc networks. In: Proc. of ACM MSWiM. (2000)
9. Lou, W., Wu, J.: On reducing broadcast redundancy in ad hoc wireless networks. IEEE Transactions on Mobile Computing **1** (2002) 111–123
10. Peng, W., Lu, X.: On the reduction of broadcast redundancy in mobile ad hoc networks. In: Proceedings of ACM MobiHoc. (2000) 129–130
11. Qayyum, A., Viennot, L., Laouiti, A.: Multipoint relaying for flooding broadcast message in mobile wireless networks. In: Proc. of HICSS. Volume 9. (2002) 298
12. Sucec, J., Marsic, I.: An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks. CAIP Technical Report 248, Rutgers University (2000)
13. Cartigny, J., Simplot, D., Stojmenovic, I.: An adaptive localized scheme for energy efficient broadcasting in ad hoc networks with directional antennas. IEEE Transactions on Commuincations (2003)
14. Choudhury, R.R., Vaidya, N.H.: Ad hoc routing using directional antennas. Technical report, Dept. Electrical and Computer Engeineering, University of Illinois at Urbana Champaign (2002)
15. Hu, C., Hong, Y., Hou, J.: On mitigating the broadcast storm problem with directional antennas. In: Proc. of IEEE ICC. (2003)
16. Shen, C.C., Huang, Z., Jaikaeo, C.: Directional broadcast for ad hoc networks with percolation theory. Technical report, Computer and Information Sciences, University of Delware (2004)
17. Wu, J., Dai, F.: Broadcasting in ad hoc networks based on self-pruning. Proc. of IEEE Infocom (2003)
18. Das, B., Sivakumar, R., Bhargavan, V.: Routing in ad hoc networks using a spine. In: Proc. of IEEE IC3N. (1997) 1–20
19. Alzoubi, K.M., Wan, P.J., Frieder, O.: Distributed heuristics for connected dominating sets in wireless ad hoc networks. Journal of Communications and Networks **4** (2002) 22–29
20. Liao, W.H., Tseng, Y.C., Sheu, J.P.: GRID: A fully location-aware routing protocol for mobile ad hoc networks. Telecommunication Systems **18** (2001) 37–60
21. Dai, F., Wu, J.: An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. IEEE Transactions on Parallel and Distributed Systems **15** (2004) 902–920
22. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: Proceedings of MobiHoc. (2002) 194–205

Trellis-Based Virtual Regular Addressing Structures in Self-organized Networks

Julien Ridoux[1], Anne Fladenmuller[1], Yannis Viniotis[2], and Kavé Salamatian[1]

[1] LIP6 - UPMC, 8, rue du C. Scott, 75015 Paris, France
{julien.ridoux, anne.fladenmuller, kave.salamatian}@lip6.fr
[2] ECE Department - NCSU, Box 7911, Raleigh, NC 27695, USA
candice@eos.ncsu.edu

Abstract. Self-Organized Networks are multi-hop networks that do not rely on any infrastructure. Moreover, their topology is supposed to be flat since all nodes are equally in charge of address allocation and routing functions. Their time-changing topology breaks the association between Identification and Location that is present in the IP address. Moreover, mobility can not be handled easily when the addressing space is based on a tree description. In this paper we propose the use of Virtual Regular Structures to provide desired properties for Self-Organized Networks. Our approach opens new ways to build addressing spaces and enables an implementation based on trellis graphs. We propose a construction heuristic and evaluate its performance via simulations. We also analyze the robustness of the proposed approach with regards to mobility.

Keywords: Self-Organized Networks, Regular Structures, Dynamic Address Allocation.

1 Introduction

Self-Organized Networks (SON) represent a network model proposed to support spontaneous and multi-hop networking without relying on any precise and *a priori* infrastructure. SONs are a general description unifying well-known research fields such as Ad-Hoc or Hybrid[1] networks, Peer-To-Peer systems and Sensor networks. A SON is a spontaneous multi-hop network whose time changing topology depends on node arrivals in an environment without infrastructure.

In SONs, routing, addressing, location management, name resolution, all are non-trivial issues because of the node mobility. We argue in this paper that the choice of the addressing scheme has a serious impact on all these issues. An address can uniquely identify a node, inform on its localization, or it can also represent a combination of both. When it only identifies a host, the system has to ensure the uniqueness of each address and mobility has no influence on

[1] We define hybrid networks as Ad-Hoc networks that can obtain an Internet connectivity with the aid of some of their nodes.

R. Boutaba et al.(Eds.): NETWORKING 2005, LNCS 3462, pp. 511–522, 2005.

the address value. On the other hand, if position information is included in the address, the address has to be modified each time a node moves or appears in the network. This increases the cost of mechanisms used to disseminate or retrieve the association between a node's identifier and its location. The address allocation and routing problems can then be seen as a generic minimum cost query problem to execute on an infrastructure-less environment; nodes query addresses and routes.

A large suite of Ad-Hoc network protocols based on IP addressing have been designed to allow a dynamic auto-configuration of nodes and a routing protocol definition ([1],[2],[3],[4],[5],[6]). While providing straightforward compatibility with wired IP networks, this approaches inherits the hierarchical description of IP addresses applied to a flat topology. These proposals address the query problem with a flooding mechanism: improved flooding requests for route discovery and a Duplicate Address Detection (DAD) mechanism for address allocation. Geographical routing [7] answers the route query problem thanks to the nodes geographic coordinates. Their identification is decorrelated from their location, implying the need of a positioning system and a location mechanism such as GLS (Grid Location Service) [8], to realize the address queries. Peer-To-Peer systems [9] define a virtual addressing space based on Distributed Hash Tables to realize the correspondence between the node identifier and its location in the virtual space. The routing query problem is solved by the underlying layer. The approach in [10] pushes the P2P concepts into the routing layer but its addressing space (based on a tree structure) copes poorly with the dynamic nature of SONs. To avoid flooding, GLS and the Peer-To-Peer approaches introduce a virtual structure that decreases the cost of the addressing query problem. The study done in [11] shows that structured networks are helpful to introduce a coherent addressing scheme and improve routing and maintenance performance. We claim that the use of a virtual structure will give "good" topological properties to SONs by decorrelating the node identifier from its location.

Our proposal provides a topological Virtual Regular Structure (VRS) to map the addressing space, fitting the geographical node placement. We propose trellis graphs as the basic element to implement a VRS in SONs. Trellises can be used for both creating the address space and finding routing paths. Thanks to their redundant structure, they provide an implicit multi-path construction and are robust toward frequent node arrival and departures.

The key point of our proposal is to design an addressing space that is set up in a "constructive" and local manner. This set up avoids the need of a global view of the network, difficult to obtain in a SON. Such a construction adapts the size of the addressing space to the number of nodes present in the SON while providing high scalability by definition. Our proposal avoids address allocation conflicts, without widespread flooding, allowing our solution to scale large networks by construction. In terms of routing, our virtual structure includes redundancy that provides an implicit multi-path definition that is an essential property for SONs.

In the rest of the paper, section 2 describes the general use of regular structures and the trellis implementation. Sections 3 details the addressing space construction. Section 4 contains analysis in terms of robustness of our approach in the presence of mobility. Section 5 presents simulations showing the feasibility of constructing a regular VRS.

2 Generic Virtual Structure

2.1 Generic Description

We propose the use of VRS in SONs to decrease the cost of the solution to the generic query problem described. This regular structure has to possess properties that facilitate both addressing and routing. This regular structure is repeated to integrate all the nodes of a SON. In order to allow a communication between these regular structures, they have to be connected to each other. To keep the same properties induced by the regular structure, we propose to connect them in a recursive manner. Some of the nodes act at different levels of recursion and provide the connectivity between those levels. These nodes are called "Structure Heads", represented in grey and black color in figure 1(a).

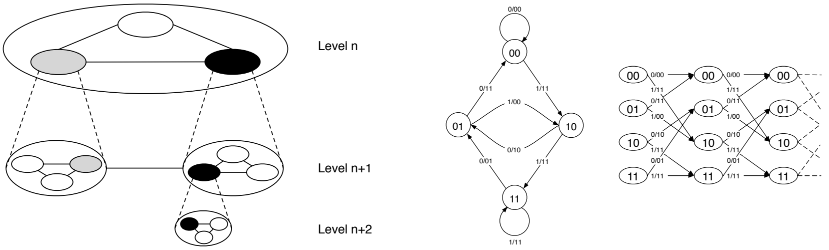
To be able to route in such a VRS, each regular structure must possess an address that identifies it and gives the recursion level in which it is present. Moreover, each node of a regular structure must possess at least one address relative to the structure and all nodes in a regular structure must be able to communicate to each other, through a path in the physical layout.

The address of a destination contains the path to reach it in the whole virtual structure. This path is composed of the list of Structure Heads of each regular structure a packet has to go through. Locally to a structure, the routing decision is taken compared to the address of the next Structure Head. The knowledge of which regular structure to forward packets to is given by the address identifying each regular structure. In order to realize this routing mechanism, each link present in the virtual structure must correspond to a physical one.

2.2 Realization Based on Trellis Graphs

We chose to use trellis graphs to implement the general concepts we just described. Trellis graphs allow robustness toward mobility of nodes thanks to the redundancy they imply. Their algebraic definition allows an easy and natural multi-path routing. To the best of our knowledge, trellis graphs have been used only in [12] at the networking level. Their application in providing a VRS is novel. A trellis graph can be generated by a Convolutional Encoder [13]. Figure 1(b) represents such a trellis graph and its representation as a Finite State Machine (FSM).

The FSM states are labeled by binary values indicating the current bits stored in the encoder memory. The transitions are labeled by binary values (*e.g.*, 0/11) indicating the entry given to the FSM (0) and the corresponding output code (11). The same information is represented on the corresponding trellis. From state 00, the FSM changes to state 10 if the new input value is 1 and stays in



(a) A recursive regular structure (b) FSM and Trellis representation

Fig. 1. Virtual Regular Structures

state 00 if the input value is 0. Words are encoded by reading them in Least Significant Bits order.

A trellis is a connected graph, repeating a set of vertices, all having the same degree. The vertices of the trellis represent the different FSM states and their edges the FSM transitions. The representation of the FSM state changes for a given input sequence corresponds to a path in the trellis. In the following we use the FSM and the trellis illustrated by figure 1(b) to detail the mechanism used for addressing and routing.

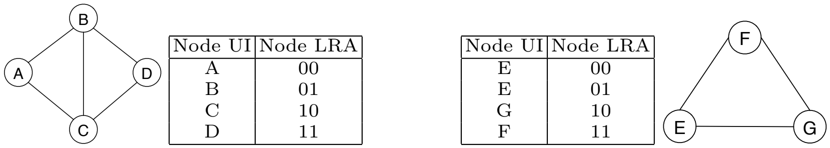
3 Virtual Address Space Based on Trellises

In order to precisely describe the way we organize the Virtual Regular Structure, we state some hypotheses first. We consider a Self-Organized Network built from scratch, where node arrivals are sequential. The physical topology created by this arrival process can not be predicted and we assume no particular property on it. We suppose that no infrastructure is present either, to help the addressing or routing mechanisms[2]. We make the hypothesis that each node joining the network possesses a unique Universal Identifier (UI). The definition of this identifier is outside the scope of this paper; the reader can refer to the large amount of literature in different fields such as Peer-To-Peer systems [9]. To allow compatibility with the existing IP network, an IP address could be used as the node UI, as long it is guaranteed to be unique. Our approach can then be implemented as a routing daemon in any Ad Hoc routing protocol, providing full compatibility with existing systems.

3.1 Description of a Single Trellis

The vertices of a trellis graph represent the location of nodes within the virtual structure we want to set up. The FSM associated to the trellis is known *a priori*

[2] An existing infrastructure can be present and would ease the construction of the VRS in a transparent way. Here we consider the general case without any infrastructure.



Node UI	Node LRA
A	00
B	01
C	10
D	11

(a) 4 nodes mapping

Node UI	Node LRA
E	00
E	01
G	10
F	11

(b) 3 nodes mapping

Fig. 2. Basic Physical Topologies and Association Tables

by all the nodes composing the network. The state of the FSM associated to a node is called its Local Relative Address (LRA). Since the identifier and location of a node are decorrelated, we need to introduce an address Association Table to map the node UI to its Local Relative Address. Figure 2 gives two simple physical network topologies to illustrate this mapping where capital letters represent the nodes UI. The topology of figure 2(a) is composed of four nodes each associated to one of the FSM states.

In figure 2(b), since the number of physical nodes is less than the number of states of the FSM, node E is represented twice, *i.e.*, E is given two addresses in the regular structure. As shown in this example, each trellis possesses a given number of addresses and one or several of these addresses are allocated to a physical node. The resulting Association Table representing neighbor nodes depends on their arrivals. Different node arrival sequences, resulting in the same physical topology could produce different node UI to FSM state mappings, and different numbers of Local Relative Addresses of a node. The Association Table creation is shortly described in section 3.3. In the examples given in figure 2, the paths to nodes are their LRA. If node A needs to communicate with node D, it will use D's LRA (11). By construction, each edge of the trellis corresponds to an existing physical link. A can then find a path in the trellis to reach D.

3.2 Spanning the Entire Network

Recursion is introduced by creating trellises of trellises. This recursive construction is our proposed solution to span the entire network by repeating the same regular structure. Figure 3 shows trellis graphs T_3, T_4 and T_2 mapping respectively the nodes $\{A, B, C, D\}$, $\{E, F, G, H\}$ and $\{I, J, K\}$.

In order to allow all the nodes of the network to communicate, the trellis graphs T_2, T_3 and T_4 are interconnected in a recursive manner. Figure 3 presents a VRS of two levels of recursion, where T_1 associates T_3 to its states 00 and 10, while it associates T_4 to its states 01 and 11. T_0 and T_1 can then be described as “trellises of trellis graphs”.

The VRS structure we describe aims to provide both an addressing space and a routing protocol. But trellis graphs belong to the virtual space. To communicate from one trellis to another, the communication has to be ensured by the physical nodes interconnecting the two trellises. To realize all the operations our mechanism needs, trellis graphs are represented at any level of recursion by some of the physical nodes that permit this interconnection. The nodes associated to

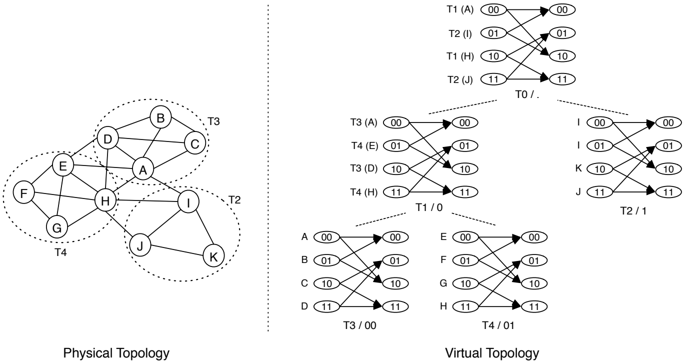


Fig. 3. Spanning the Network

the lowest and highest value of the FSM states are chosen to represent their trellis in higher recursive levels. These states are selected because they own a loop edge in the FSM making easy to represent the recursive interconnection. We name these two connection nodes the Trellis Heads. In figure 3, T_3 is represented in T_1 by its Trellis Heads A and D. In the same manner, T_4 is represented by nodes E and H in T_1.

As we mentioned in section 2, we need to define a Trellis Prefix (TP) to identify each regular structure. By definition each trellis of size 2^L combines $2^{(L-1)}$ sub-trellis. Each trellis T attributes a prefix p written as a bit string of size $2^{(L-1)}$ to the nodes representing a common sub-trellis. p is the recursive concatenation of T's TP with the bit string attributed by T to a sub-trellis. The highest trellis in the recursive hierarchy possesses a prefix of size 0. T_4's TP (01) in figure 3 illustrates this concatenation of T_1's TP and T_4's prefix in T_1.

Once the trellis structure is established, all the states of each trellis composing the VRS are associated to a physical node. It is then possible for a node to retrieve the path to any destination in the VRS. This path, called the Relative Address (RA), is a bit string composed of the concatenation of Local Relative Addresses. For each trellis a packet goes through to reach the destination, one LRA is concatenated to the Relative Address. The LRA added in each trellis is the LRA of the last node the packet goes through before being forwarded in another trellis. The RA is the sequence of "exit vertices" of each trellis to go through to reach the destination. Since the RA corresponds to a path, a destination RA is different for two distinct source nodes.

As an example, consider K's Relative Address in figure 3. Let suppose that F and J need to reach K. For J, the case is trivial. J and K belong to the same trellis, they share the same Association Table, and J knows K's RA as 10. From F's point of view, K's RA will be 00.11.11.10. That means E's LRA in T_4 (00), H's LRA in T_1 (11), J's LRA in T_0 (11), K's LRA in T_2 (10). It is important to notice, that this is not the only possible RA for K from F's point of view.

Since we introduce redundancy for multi-path properties, and because the RA is bound to the path, 11.00.01.10 would have been another possible path.

3.3 Heuristic for Trellis Construction

Establishing an optimized trellis-based VRS, as we just described, is intuitively the biggest difficulty of our approach. In a more formal manner, the construction of the optimal trellis-based VRS is an NP-Complete problem as we have shown in [14]. The optimality criterion defined involves a minimum number of trellis and a minimum number of recursive levels to cover the whole topology. Since finding an optimal trellis VRS is an NP-Complete problem for a given topology, we have proposed a heuristic to set it up, based on an incremental arrival of nodes. In this way, our heuristic builds the top trellis of the VRS first and adds new, lower level trellises as nodes join the SON. When a node joins a network, it tries to join the existing VRS by being inserted in an existing trellis. If no possible position is available, because of physical connectivity constraints, the arriving node creates a sub-trellis with its neighbor. This process ensures that the heuristic is able to configure any new node joining the network. As we will describe later, our proposal allows an implicit merging of trellises when nodes do not appear sequentially. More details about this heuristic are given in [14].

3.4 Packet Forwarding

As nodes join the network, they register recursively their positions in the virtual topology until reaching the highest level trellis. This registration is similar to the one proposed in GLS in the case of geographic routing. The route request from a source node S is propagated to higher levels of hierarchy until reaching the first node L possessing an entry for the requested destination D. By concatenating the path from S to L and from L to D, the source node retrieves a complete path to the destination.

Each forwarding node realizes a routing decision based on the address of the destination in the topology (present in the packet header), the FSM defining the trellis and its own location in the VRS. As the address of the destination nodes represents a sequence of states in the FSM, this process gives a direct routing decision on a per-trellis basis, making it possible to forward the packets along the structure. Because of space limitation, we do not describe here the mechanisms used for packet forwarding in detail. A complete description can be found in [14].

4 Robustness Analysis

Mobility is arguably the biggest challenge SONs have to face in regards to addressing and routing. Our approach decorrelates node identifier and position, which provides the basis to handle mobility. In order to position our work toward the different mobility aspects, we define four mobility categories. The first one is the join and leave processes of nodes. The second kind of mobility, “slow individual mobility", is characterized by a continuous node mobility, where the node movement can be tracked gradually by its neighbors. The third aspect can be defined as "fast individual mobility" and does not allow the tracking of the mobile node. The node is present continuously in the network but its neighborhood changes completely between two given instants. The last category of mobility we consider here is group mobility. This type of mobility leads to network splits and mergers, that are the aspects any addressing space (except the geographic one) has difficulty to handle. Combining these different kinds of mobility creates numerous scenarios that can not be described here.

4.1 Slow Mobility

Because of the effect they have on our scheme, we combine the join and leave process with the individual slow mobility of nodes. The join process of a node is handle by the construction process of the VRS. When a node moves or leaves, it produces broken links in the trellis graphs it was present. Because of the introduction of redundancy, the structure remains robust and is able to find alternative paths to route packets that were intended to go through the missing node. Since a trellis is a representation of a Convolutional Encoder, the analogy with error recovery is straightforward. While the number of active links remains over the capability threshold of the trellis, there exists a path (obviously longer than the shortest one) to reach an existing destination. When the number of active routes falls under the threshold, the trellis structure does not allow routing and needs to be reconstructed.

The reconstruction of a trellis is a local procedure. This is arguably the best advantage toward mobility, since a trellis reconstruction will not have an impact on the entire network. When the number of routes of a trellis falls under the threshold, the nodes remaining in the trellis restart a bootstrap process as if they were just joining the network. Since their UI remains the same, the protocol is able to guarantee reception of all packets they have to receive, at a cost of some retransmissions.

4.2 Fast Mobility

Fast Individual Mobility of a node changes its neighborhood and therefore its topology knowledge changes very fast. In order to be able to handle this kind of behavior, our approach must be able to configure trellis graphs, register addresses and route packets at a higher speed than the node movement. Due to the complexity of the trellis set up, and the possibility for having continuous trellis reconfigurations, we do not claim that our trellis-based VRS is able to handle such node behavior. Nevertheless, we think that our approach will cope with most realistic situations.

4.3 Group Mobility

Because our VRS is based on a neighbor node construction, two new-formed network partitions, that result from a network split, remain coherent. They form two independent recursive subtrees of the VRS. For each network partition, some nodes representing the missing trellis graphs would have disappeared in different recursion levels. This situation is handled as a successive sequence of node departures. The trellis graphs forming the new border with the other partition may have to be reconfigured. Again, the split operation is handled locally and does not influence the entire network. If a higher level trellis completely disappears, the prefixes of the remaining trellis graphs have to be updated by removing the bit string corresponding to its Trellis Prefix.

A network merger can be detected when, for example, two configured nodes M and N meet and the request for a route to each other does not give any result for each of them. A network merger has no influence on the addressing space, or on the routing topology thanks to the recursive property of our structure. In other words, the VRS of a prior network is placed "under" the other one.

In order to guarantee that the whole VRS remains coherent, M has to become a part of N's trellis. At the same time, in order to allow packets to be forwarded between N and M's networks, M has to be present at the top trellis of its partition. These conditions imply that, M chooses to join N's trellis because its VRS contains fewer recursive levels of trellis graphs. If M is a trellis head present at the highest level of hierarchy, it joins N's trellis, and the network merger procedure is finished. If this is not the case, M starts a trellis head permutation procedure to become a trellis head present at the highest level of recursivity.

Our approach takes advantage of the redundant and recursive properties of the VRS to handle broken routes, slow node mobility and group mobility. Even though we did not describe the mechanisms involved precisely, because of paper length limitation, we show the advantages of such properties for SONs.

5 Evaluation and Simulations

It is always possible to build the VRS containing all the network nodes with recursive trellises. Figure 4 is an example of a resulting trellis-based VRS construction. Since such construction is an NP-Complete problem for an existing topology, it is important to simplify the construction as much as possible, while maintaining acceptable performance in terms of computation load on the nodes for the routing, minimization of the information storage required for our VRS or matching of the physical minimum route and the path in the VRS to the destination.

Since we use only local information to optimize routing, we expect the trellis-based shortest path to always be in the same order of magnitude as the physical shortest path. We computed the difference of the path length between the physical shortest path and the path produced by the trellis-based VRS. We do not present the obtained simulation results in this paper, as the number of simulation runs was not sufficient to have a real statistical meaning. So far these first encouraging results give us the indication that the path length does not increase exponentially.

The computation required to route packets along the VRS mostly depends on the length of the path between nodes. The longer the path in the VRS, the more processing time is required to compute it, since more nodes are involved

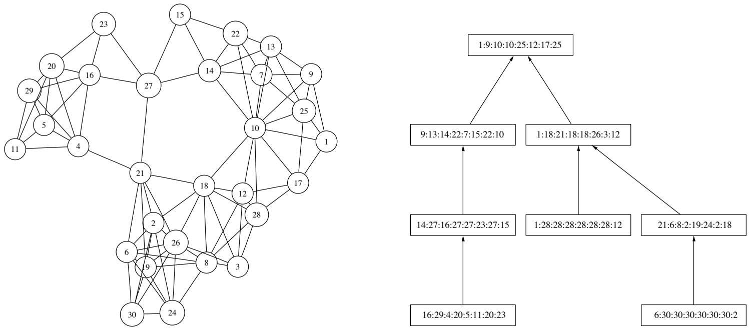


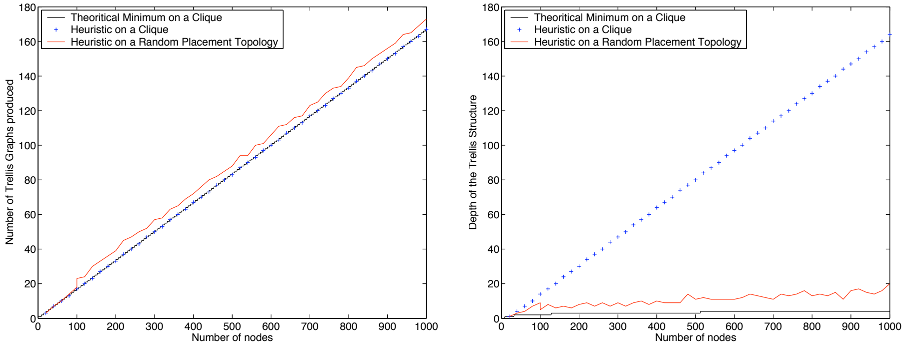
Fig. 4. 30 nodes Random Placement Topology and its VRS

in encoding bit strings representing addresses. Statistically, it seems intuitive that the depth of the VRS will have an impact on the average length of the path between nodes. The depth of the VRS can give us some insight on the performance in terms of average routing processing load for a given topology.

Each trellis stores information on each association (LRA, UID) of its own trellis and of all the trellises of longer prefixes below it. The size of the trellis is fixed, and several LRAs can be associated to one UID. So, in order to optimize the amount of information stored in the VRS, we need to minimize the number of trellises and to create the shortest structure possible. This is what our heuristic tends to do, by limiting the creation of new trellises [14]. Figure 4 shows the resulting VRS thanks to our heuristic on a random placement topology.

As a result of this analysis, we can deduce that the depth of the virtual structure and the number of trellises has an impact on the performance of our approach. Although our heuristic tends to minimize both metrics, it can not be guaranteed they will both be optimized. One major characteristic of our proposal is that, although we are sure a VRS can be built, we can not predict its resulting shape. As a matter of fact, for a given topology, several VRS can be built, depending on the node arrival sequence: the number of trellises as well as the maximum number of recursive trellises (depth of the VRS) can differ. It is thus important to understand which parameter will impact performance the most.

In order to evaluate the construction heuristic we proposed, we generated two kinds of topologies. One is a fully-connected topology, where all nodes are 1-hop neighbors. The second one is a randomly placed node topology on a square area of size 1000x1000 meters. Each wireless node possesses a transmission range of 250 meters, a widely used value for simulators (as in NS-2 for example). In these simulations, we studied the construction of the VRS that is the core of our proposal. We tested the heuristic we propose by using a trellis of size 2^3. This parameter remains constant for all of the following simulation results. To evaluate simulation results, we computed the theoretical minimum number of trellis and the minimum depth of the recursive structure one can obtain for a given number of nodes.



(a) Number of constructed Trellis (b) Depth of the Trellis-based VRS

Fig. 5. VRS Construction characteristics for different topologies

5.1 Number of Trellises

Figure 5(a) shows the number of trellis graphs produced on the two kinds of topology and the theoretical minimum. On a topology where all nodes are able to communicate in 1 hop (on a clique), our heuristic produces the minimum possible number of trellises. This indicates that the heuristic behaves correctly. This topology is nevertheless an ideal case, as the more nodes are connected, the easier it will be to fill each trellis with new nodes and thus reduce the number of new trellises. For more realistic topologies, where nodes are placed randomly, the upper curve shows that our heuristic remains close to the optimum number of trellises.

5.2 Depth of the VRS

Figure 5(b) shows that the depth of the structure constructed on such a fully-connected topology is very high. This is explained due to the algorithm of our heuristic. Since there is no physical topology constraint in this situation, the first trellis tested for an available position will accept the new node. The other possible trellises are never tested and this leads to a very high depth. We can also see that the curve representing the same situation increases but stays reasonably close. It shows that in a more realistic situation our heuristic behaves pretty well, even with more physical connectivity constraints. Optimizing both the number of trellises and the depth of the structure is really difficult, and can be the next improvement to this heuristic.

6 Conclusions

In this paper we have proposed the use of virtual Trellis structures to (a) organize the addressing space of a SON in a structured fashion, and, (b) to perform data routing in such an environment. Trellis structures are well known in classical communication and information theory, but their application in SON is novel.

We strongly believe that introducing virtual structures for addressing space that are not based on trees will open new research areas. Thanks to the redundancy and recursion introduced in the trellis-based VRS, we provide a distributed dynamic addressing scheme, with the following advantages: localized operations, no size limit in the addressing space, a built-in multi-path routing structure, computed locally, and robustness to various kinds of mobility. The limited size of the trellis and the definition of local operations allow the addressing and routing to be highly scalable. The main drawbacks of our approach are (a) routes are suboptimal, in terms of number of hops, and, (b) control overhead is not fairly distributed among nodes.

References

1. R. Wakikawa, J.T. Malinen, C.E. Perkins, A. Nilsson and A.J. Tuominen: Global Connectivity for IPv6 Mobile Ad Hoc Networks. Internet-draft, IETF (2002)
2. D.B. Johnson and D.A. Maltz: Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile Computing, Kluwer Academic Publishers **353** (1996)
3. S. Nesargi and R. Prakash: MANETConf: Configuration of Hosts in a Mobile Ad Hoc Network. In: Proc. of IEEE INFOCOM. (2002)
4. H. Zhou, L.M. Ni and M.W. Mutka: Prophet address allocation for large scale MANETs. In: Proc. of IEEE INFOCOM. (2003)
5. C.E. Perkins, E.M. Royer and S. Das: Ad Hoc On Demand Distance Vector (AODV) Routing. RFC 3561, IETF (2003)
6. C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum and L. Viennot: Optimized Link StateRouting Protocol. RFC 3626, IETF (2003)
7. M. Mauve, J. Widmer and H. Hartenstein: A Survey on Position-based Routing in Mobile Ad Hoc Networks. IEEE Network Magazine **6** (2001) 30–39
8. J. Li, J. Jannotti, D. De Couto, D. Karger and R.Morris: A Scalable Location Service for Geographic Ad Hoc Routing. In: Proc. of ACM MOBICOM. (2000)
9. I.Stoica,R.Morris,D.Karger,M.F.Kaashoek,H.Balakrishnan: Chord:A Scalable P2P Lookup Service for Internet Applications. In: Proc. of ACM SIGCOMM. (2001)
10. A.C. Viana, M.D. Amorim, S. Fdida and J. F. Rezende: An Underlay Strategy for Indirect Routing. ACM Wireless Networks **10** (2004) 747–758
11. M.Castro, M.Costa and A.Rowstron: Should we build Gnutella on a structured overlay? In: HotNets-II. Cambridge, MA, USA. (2003)
12. S.D.Nikolopoulos, A.Pitsillides, and D.Tipper: Addressing Network Survivability issues by finding the K-best paths through a Trellis Graph. In: Proc. of IEEE INFOCOM. (1997)
13. S.Lin, D.J.Costello: Error Control Coding: Fundamental and Applications. Electrical Engineering Series. Prentice-Hall (1983)
14. J.Ridoux, A.Fladenmuller,K.Salamatian and Y.Viniotis: Beyond the Tree Structure: a new way to configure nodes in SONs. Technical Report, UPMC-LIP6 (2004) http://www-rp.lip6.fr/~ridoux/Publications/TR_BeyondTrellis.pdf.

Network of Shortcuts: An Adaptive Data Structure for Tree-Based Search Methods

Andrea Bergamini[1] and Lukas Kencl[2]

[1] Ecole Polytechnique Fédérale de Lausanne (EPFL),
1015 Lausanne, Switzerland
andrea.bergamini@epfl.ch

[2] Intel Research, 15 JJ Thomson Avenue, Cambridge,
CB3 0FD, United Kingdom
lukas.kencl@intel.com

Abstract. In this work we present a novel concept of augmenting a search tree in a packet-processing system with an additional data structure, a *Network of Shortcuts*, in order to adapt the search to current input traffic patterns and significantly speed-up the frequently traversed search-tree paths. The method utilizes node statistics gathered from the tree and periodically adjusts the shortcut positions.

After an overview of tree-search methods used in networking tasks such as lookup or classification, and a discussion of the impact of typical traffic characteristics, we argue that adding a small number of "direct links", or shortcuts, to the few frequently traversed paths can significantly improve performance, at a very low cost. We present a shortcut-placement heuristic, compare our method to a standard caching mechanism and show how the use of different levels of aggregation in a search tree enables to achieve similar results with much fewer entries.

1 Introduction

1.1 Networking and Search Trees

The explosive Internet growth requires complex tasks to be processed ever faster by the intermediate systems such as routers or firewalls. Optimization of these processes is necessary, yet often impossible a-priori, as the optimum often depends on the particular traffic mix. Run-time self-reconfigurable devices offer a solution with several advantages, i.e. optimized resource utilization, improved performance and ease of programming and management.

In networking systems, several targets for dynamic reconfiguration can be identified: resources re-mapping [1], codepath adaptation or *data structure* adaptation. Data structures comprise e.g. forwarding tables [2] and classification rule bases [3], which can be very large (e.g. 100k prefixes or 32k rules [3]) and the search speed is a major performance factor. As the search usually requires a number of memory accesses, and memory latency lags behind processor and link transmission speeds growth, optimization of data structures is critical. Typically

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 523–535, 2005.

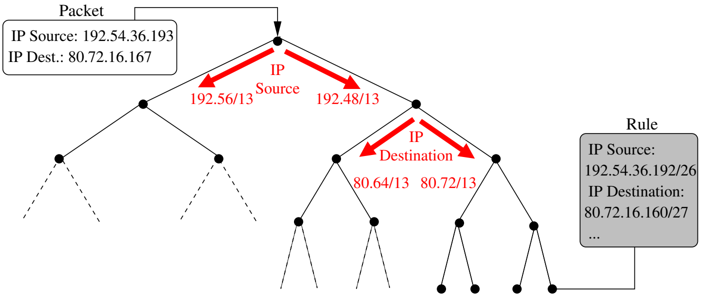


Fig. 1. Example of tree-based packet classification, as in e.g. [3]. Here, the root test is based on the IP source address, and the process is repeated at each node, where the test may be based on different, possibly multiple, packet fields

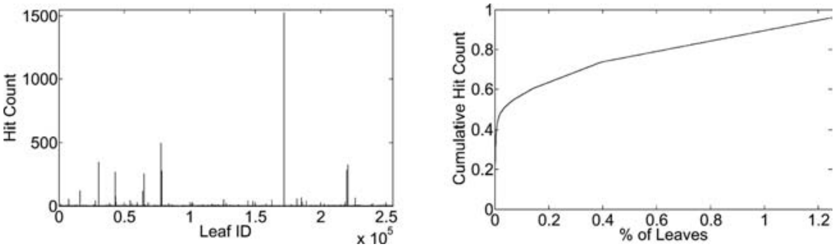


Fig. 2. *Example:* few tree paths are frequently searched. Tree depth 9, degree 4, generated packet flows conform to a measured length distribution (see Subsection 3.1). *Left:* Hit count distribution at tree leaves. *Right:* Cumulative Hit count. 90% of the traffic concentrated in less than 1% of the top searched paths

built based on the prefix tables or rule-bases and optimized for the worst case traffic, the data structures may be grossly sub-optimal for the actual traffic.

Various techniques are used to perform these optimizations [3, 4, 5, 6, 7], e.g. hash tables, compressed bitmaps or tree-like structures. HiCuts [4] and HyperCuts [3] deploy *search trees* to perform packet classification, the process of identifying the highest-priority rule applying to a packet. Both algorithms use heuristics to guide the tree construction, trading-off memory space and speed. At each node, a test is performed on the subset of the packet fields that narrows most the amount of rules that may apply to the packet (see Fig. 1), until a leaf is reached. The rules can be a combination of a fixed match, prefix match or a range match over various fields of the packet header, and may span multiple tree leaves.

The input, Internet traffic, is typically studied as aggregation of *flows*, sequences of packets sharing the same 5-tuple flow ID[1]. Previous studies [8] confirmed that some of the flow parameters are correlated (e.g. rate and size) and

[1] The 5-tuple flow ID is defined as: IP source and destination addresses, IP protocol number and TCP/UDP source and destination port numbers.

that small flows (mice) represent the majority, yet most of the traffic (in bytes) is concentrated in few big flows (elephants). In traffic engineering [9] it is common practice to identify flows that capture most of the traffic at a certain point in time and treat those differently to optimize resources utilization. This relies on such sets being persistent at least on a small time scale (shown to hold for flow volumes [10]). Temporal locality, the likelihood of a recently observed flow being referenced in the near future, is strongly present in the Internet traffic [11]. Critically, in search trees, the bias in flow lengths (many mice vs. few elephants) is reflected as *imbalance in the tree traversal patterns*, as a vast majority of searches typically end in a tiny subset of tree leaves (see example simulation in Fig. 2).

1.2 Objective: Adaptive Tree Search Optimization

In this work, we focus on run-time optimization of *tree-based* search data structures [3, 4]. It is critical to minimize the depth of the search tree to minimize the number of memory accesses. As some parts of the tree are likely to be traversed more frequently, we ask: *"Assuming we knew the traversal pattern, how could we optimize the tree for this pattern?"* The assumption is realistic, as it is possible to gather such statistics at line rate [12, 13]. The above observations about Internet traffic, such as the persistence property, motivate optimizing the data structure for the current time frame using the tree hit statistics from the previous time frame. Splay trees [14] and randomized binary search trees [15], are known examples of adaptive tree data structures, however, not applicable in this environment.

A common way of exploiting temporal locality is a *cache* of search keys, e.g. flow IDs. The Least Recently Used (LRU) cache replacement scheme has been shown to be near-optimal for networking workloads [16]. A cache cannot exploit information from the search data structure itself, so if the data structure is updated, there is a discrepancy between the contents and the cache must be flushed, which negatively affects the hit rate. The cache works with fine-grained fixed match keys (e.g. a 5-tuple flow ID), preventing aggregation of keys classified the same way, which can be exploited by a Denial of Service (DoS) attack by flooding with bogus entries. To achieve an acceptable hit rate, the cache needs to have a considerable size (e.g. 12,000 entries [17]) for a typical networking workload. With the increasing key size, this may become prohibitively large.

We propose a novel method to augment the existing search tree with an additional data structure, a *Network of Shortcuts (NoS)*, along the frequently traversed paths. The objective is to capture a large fraction of traffic and reduce the total number of memory accesses per time frame. The hierarchical aggregation of search keys in subsequent tree nodes is preserved and thus the method scales well with the number of distinct search keys (e.g. flow IDs). Heuristics to determine where to place the shortcuts are presented.

The combination of a search tree and statistics-based shortcuts may recall the *Huffman tree* [18]. But a pure Huffman tree construction would alter the node tests, and thus would not reflect the rule base structure.

Both caching and shortcuts reduce the number of memory accesses in the average case, possibly affecting the worst case scenario. The affects on the worst case can be limited by constraining a number of shortcuts along a single path, as well as by adjusting the method parameters or disabling it altogether, should the current traffic patterns not be favorable. We demonstrate that a controlled adjustment of the worst case (i.e. keeping the worst case bounded) results in massive gain in the average case. The tree-based packet classification is a working example, but the shortcut method can be applied to various tree-search methods.

The paper is organized as follows: in Section 2 we present the shortcut method and in Section 3 the performance results and a LRU cache comparison using generated and real traces. Section 4 describes a possible implementation of the method, and in Section 5 we discuss open problems and add concluding remarks.

2 The Network of Shortcuts (NoS) Method

2.1 Assumptions and Notation

1. We assume a tree-based search method, like HyperCuts [3], to be deployed. Backtracking, or any other mechanism that may alter the way the tree is traversed, must not be used.
2. We assume that statistics (hit count) about *every* node in the tree are periodically available. Efficient gathering of such statistics is possible [12, 13].
3. The decision at each tree node has a *cost* in terms of memory accesses (e.g. pointers to be fetched from the memory), which we assume equal to 1.

Let $F(u)$ be the periodically measured hit-count at node u, $f(u)$ the fraction of traffic at node u, $f(u) = \frac{F(u)}{F(root)}$, d the tree node degree and l the tree depth.

Definition 1. Given a tree $T = (V, E)$, a *shortcut* is an edge $e \notin E$ that connects a non-leaf *Starting Node* $SN \in V$ with its descendant in the tree T (as in Fig. 3(a)).

A shortcut is specified by a list of nodes it bypasses: shortcut $SC = (a, b, c, d, e)$. Fig. 4 presents the search before and after the placement of a single shortcut. A shortcut at a starting node SN enforces the search to test first the path using

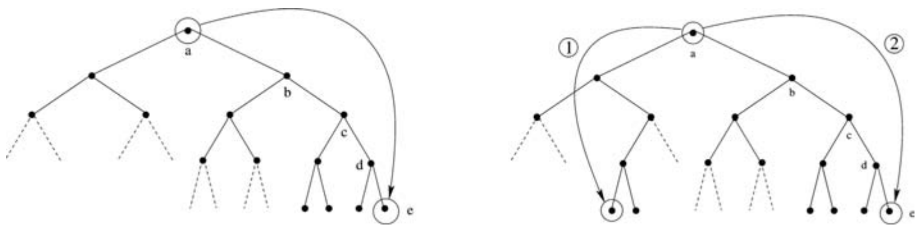
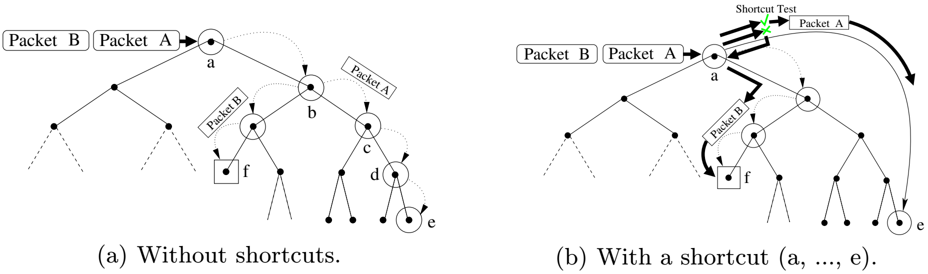


Fig. 3. *Left:* A shortcut is an additional edge between two nodes along a tree path. *Right:* A multi-shortcut. Single-shortcuts (1) and (2) are combined



(a) Without shortcuts.

(b) With a shortcut (a, ..., e).

Fig. 4. In 4(a) there are no shortcuts. At each node, packets A and B perform a test which direction to take, ending in leaves e and f respectively. In 4(b), both packets first test the shortcut. Packet A succeeds and shortcuts to e directly, whereas B fails and must retrieve the pointer along the normal path, incurring cost α

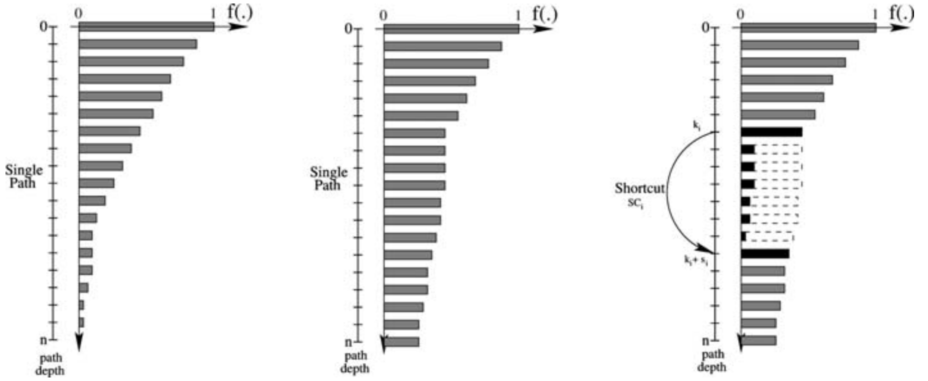


Fig. 5. *Left:* Traffic along a generic path, *center:* Traffic along a heavily-hit path, *Right:* Effect of placing a shortcut: packets skip intermediate nodes

a shortcut, and only in case of failure to continue through the search tree. Note that this requires the ability for a shortcut $SC = (a, b, c, d, e)$ to compress tests at nodes b, c and d into one test stored at node a. When a packet fails to take a shortcut, a *cost* $\alpha \geq 1$ (in terms of memory accesses) is incurred, as an additional read may be required to read the child pointer. Fig. 5 shows the effects on the traffic $f(.)$ of a single-shortcut. The notation $SC_i(m : l)$ means "from the m_{th} node to the l_{th} bypassed by the shortcut". The notion of a shortcut is a logical abstraction, for practical implementation issues please refer to Section 4.

Multi-shortcut A multi-shortcut MSC is a set of single shortcuts with a common starting node (e.g. $MSC = \{SC_1, SC_2\}$, $SC_1 = (a, b, c, \ldots)$ and $SC_2 = (a, f, g, \ldots)$, as in Fig. 3(b)). Introducing multiple shortcuts allows to increase the fraction of traffic making use of the SN. The memory access *cost* is independent from the number of shortcuts if $\forall MSC$, $MSC = \{SC_1, SC_2, \ldots\}$, $SC_i(2) \neq SC_j(2)$, $\forall\ i, j : i \neq j$, i.e. there is at most one shortcut per outgoing edge to test. Relaxing this constraint would allow more shortcuts to be placed but it could also increase the number of memory accesses required at SN.

2.2 Local Shortcut Gain Functions

Let k_i be the SN and s_i be the length of a shortcut SC_i, placed between nodes k_i and $(k_i + s_i)$ (see Fig. 5). The number of memory accesses along the path $k_i \ldots (k_i + s_i - 1)$ *before* placing the shortcut is:

$$BMA_i = \sum_{j=0}^{s_i-1} f(k_i + j) = f(k_i) + \sum_{j=1}^{s_i-1} f(k_i + j), \tag{1}$$

where BMA stands for "Before Memory Accesses". When the shortcut is added we obtain:

$$AMA_i = \underbrace{f(k_i + s_i) + \alpha[f(k_i) - f(k_i + s_i)]}_{(A)} + \underbrace{\sum_{j=1}^{s_i-1} [f(k_i + j) - f(k_i + s_i)]}_{(B)}, \tag{2}$$

where AMA stands for "After Memory Accesses". The expression (A) refers to the number of memory accesses experienced at the SN. The first term indicates that $f(k_i + s_i)$ (traffic taking the shortcut) traverses the SN at cost 1 , and the remainder $(f(k_i) - f(k_i - s_i))$ at cost α. The expression (B) accounts for the reduced number of memory accesses experienced in the intermediate nodes. We define a *metric* to evaluate the impact of a shortcut:

Single-shortcut Local Gain: Given a single-shortcut SC_i, the local shortcut gain is defined as $G_i = BMA_i - AMA_i$. After some simple computations:

$$G_i = f(k_i)[1 - \alpha + (s_i + \alpha - 2)\gamma_i], \text{ where } \gamma_i = \frac{f(k_i + s_i)}{f(k_i)}. \tag{3}$$

A shortcut is effective if the gain of placing a shortcut is positive: $G_i > 0 \Leftrightarrow s_i > 2 - \alpha + \frac{\alpha - 1}{\gamma_i}$.

Multi-shortcut Local Gain: The local gain of a multi-shortcut of degree p is:

$$G_{iMSC} = f(k_i)\left(1 - \alpha + \sum_{l=1}^{p} (s_{il} + \alpha - 2)\gamma_{il}\right); \; \gamma_{il} = \frac{f(k_{il} + s_{il})}{f(k_i)}. \tag{4}$$

If $s_{i1} = s_{i2} = \ldots = s_{ip} = s_i$ and $\gamma_i = \sum_l \gamma_{il}/p$ is the average fraction of traffic taking a single-shortcut, the gain simplifies to:
$G_{iMSC} = f(k_i)\left(1 - \alpha + p(s_i + \alpha - 2)\gamma_i\right)$ and $G_{iMSC} > 0 \Leftrightarrow s_i > 2 - \alpha + \frac{\alpha - 1}{p\gamma_i}$. Thus, given an average fraction of traffic γ_i, the multi-shortcut achieves p times higher gain than a single-shortcut.

2.3 Shortcut Placement Strategy

The shortcut placement problem is the following optimization problem:

Given: a tree $T = (V, E)$ of variable degree and node statistics $f : V \rightarrow N$.

Find: $SC = \{SC_i\}$, a set of single- and multi-shortcuts.

Constraints (conditions to determine the validity of a set of shortcuts)**:**

$\forall i, j \; : i \neq j; \; SC_i, SC_j$ are valid $\Leftrightarrow$
$(SC_i(1:2) \subset SC_j \wedge SC_i(s_i - 1 : s_i) \subset SC_j) \quad \vee$
$(SC_i(1:2) \not\subset SC_j \wedge SC_i(s_i - 1 : s_i) \not\subset SC_j)$.

Objective function: $max \sum_i G_i$, where G_i is the local gain of SC_i.

We maximize the total gain, a sum of the local gains. As the local gains are *not* mutually independent, the optimization problem is non-linear. We conjecture that the placement problem is NP-Complete, although we could not prove it. We present a heuristic that provides good performance with few computations, however, we are unaware how close our results are to the optimum:

Shortcut Placement Algorithm: Let $Node_i$ $(i = 1, \ldots, N)$ be the i_{th} node in the tree (N nodes in total, numbered from the root, top-down, level-by-level); and let $Node_{ij}$ $(j = 1, \ldots, d)$ be the j_{th} child of the i_{th} node. Let γ_{th} and $gain_{th}$ be the thresholds that discriminates the SNs, and BSS_j the “best single-shortcut” in the j_{th} subtree. The pseudocode for the shortcut placement is:

```
for $(i = 1; i <= N; i++)$ do
  if $HitCount(Node_i) \neq 0$ then
    for $(j = 1; j <= d; j++)$ do
      $BSS_j, \gamma_j, gain_j \leftarrow BestSingleShortcut(Node_{ij})$;
    end for
    $\gamma \leftarrow \sum \gamma_j$; $gain \leftarrow \sum gain_j$;
    if $(\gamma > \gamma_{th})$ & $(gain > gain_{th})$ then
      $PlaceShortcut(Node_i, BSS_1, \ldots, BSS_d)$;
    end if
  end if
end for
```

$BestSingleShortcut(.)$ identifies in the sub-tree rooted at node $Node_{ij}$ the single-shortcut that gives the highest gain, and returns the shortcut target node, γ_j and $gain_j$. The algorithm places shortcuts in a top-down fashion and reduces the number of nodes under consideration at each step downwards. The complexity is $O(N \log_d N)$, where N is the total number of nodes.

Starting from the root allows to consider the longer shortcuts, along the few paths where the traffic curve $f(.)$ is completely flat, first. Note that changing the threshold values will affect the number of accepted SNs, and that the traffic pattern may not justify placing shortcuts at all.

Adaptation Period: The shortcut placement is re-computed periodically with period TA. Statistics collected in the previous interval are then used to place shortcuts for the following interval. The initial length of the interval is determined by the network workload and the tree size (the NoS method needs "enough" packets in the tree to obtain a meaningful description).

3 Method Validation

In this Section we validate the NoS method through simulations involving both synthetic and real traces and compare the method with a *Least Recently Used* (LRU) Cache. We picked LRU for the comparison because it performed best in our settings (we compared FIFO, RAND, LFU and LRU [16]). To make a fair comparison, since the number of shortcuts placed is variable, we vary the cache size accordingly (1 shortcut = 1 cache entry). We compare the relative gain in memory access reduction, defined as $GAIN = (\sum_i G_i)/(\sum_i BMA_i)$, and the hit rate, $HITRATE = (Traffic\, taking\ the\ shortcuts\ or\ cache)/(Total\ traffic)$.

3.1 Traffic Generator, Real Traces and Simulation Parameters

Synthetic packet traces were generated using MATLAB R7 software. The traces are characterized by a constant number of concurrent flows, Poisson new-flow arrival times and 32 bit *uniformly* distributed flow-IDs. The flow length distribution is obtained combining measured flow lengths with a tail fitted using a Pareto distribution (parameter $a = 0.9163$). The uniformity of flow-IDs is not realistic. However, as this leads to a uniform spread in the search tree, it is a good "worst-case" test for the NoS method.

Throughout this study we also used real data collected by the network monitor described in [19]. The monitored site connection is a full-duplex Gigabit Ethernet. The `dump_*` and the `mon_*` traces contain approximately 3000 and 15000 unique flow-IDs respectively. The packet source/destination addresses and source/destination port numbers (96 bits in total) were mapped into the 32 bit flow-ID used in the simulations[2].

In all the simulations, the parameters are set to: $\gamma_{th} = 0.4$, $gain_{th} = 4000$, $TA = 10000$ packets (synthetic traces), $TA = 100ms$ (real traces), $\alpha = 2$. All trees are $d = 4$, $l = 9$ (no missing nodes), except where stated otherwise. The trees are balanced, to avoid results biased due to the tree structure. At every tree node, the search space is partitioned into consecutive chunks of equal size, starting with the 32-bit space at the root. In each bar graph, the number on top of the bars is the number of shortcuts placed.

3.2 NoS Versus LRU Cache

Fig. 6 presents the results using *synthetic traces*. The NoS method achieves 4 times the cache gain in every condition given the same size and up to 50 times

[2] We combined the 10 most significant bits of the addresses with the 6 least significant bits of the port numbers=2*10+2*6=32 bits.

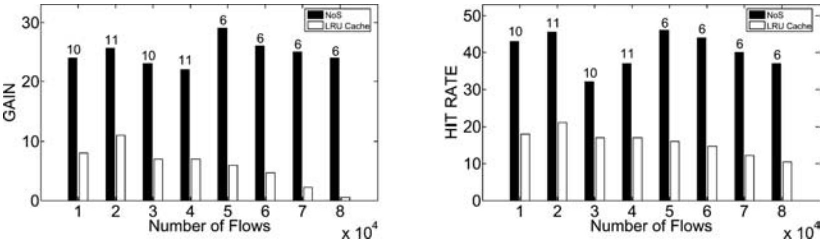


Fig. 6. NoS vs. LRU: Gain and Hitrate comparison using synthetic traces

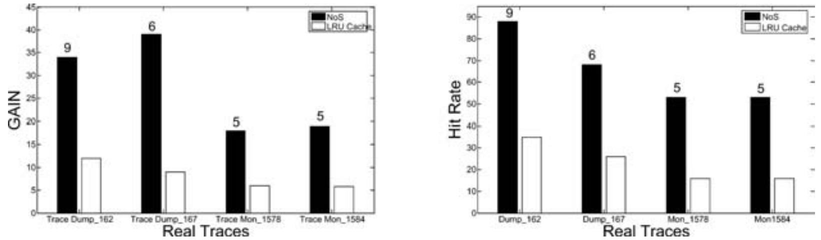


Fig. 7. NoS vs. LRU: Gain and Hitrate comparison using real traces

when the number of flows increases. This is possibly due to several factors: the aggregation of keys across shortcuts (improving scalability in the number of flows), i.e. the cache works at the flow level, whereas shortcuts target intermediate nodes, grouping multiple flows together; in addition, the shortcuts allow to capture all packets from the most populated flows, whose cache hit-rate is hurt by the less populated flows. The NoS method likewise exhibits approximately 4 times higher hit-rate.

Fig. 7 shows the results using *real traces*. For `dump_162` and `dump_167` both gain and hit rate are much higher than those observed in the synthetic case, probably due to non-uniformity in the flow-IDs. The `mon_1578` and `mon_1584` traces contained a higher number of flow-IDs, with more uniform distribution. Still, the NoS method managed to capture more than 50% of the traffic, achieving gains 3 times higher than the cache.

Worst Case and Average Case: Fig. 8 shows detailed worst case and average case statistics from one of the real traces. It illustrates that the NoS method achieves a significant improvement in the average number of memory accesses, while keeping the worst case bounded. The amount of the worst case instances is marginal (10^{-4}%). In spite of the average case improvement, fewer packets are adversely affected by NoS than by the LRU cache.

3.3 Variable Root Degree

Real-life search trees are typically of variable degree; in particular, the root degree tends to be significantly higher (e.g. 64). We present experiments with

	Tree Search	LRU Cache	NoS
Average Case	10	8.5	6.4
Worst Case	10	11	14
% Worst Case	100	75	0.00068
% Worse Off	0	75	23.12

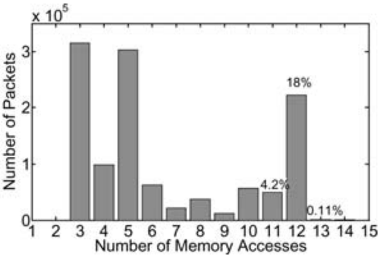


Fig. 8. Detailed results from trace dump_167. *Left:* Per-packet memory accesses. *Right:* Histogram of memory access distribution using the NoS method

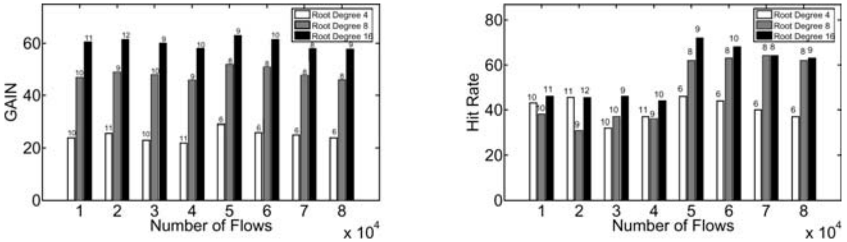


Fig. 9. NoS Gain and Hitrate comparison using synthetic traces: root of variable degree

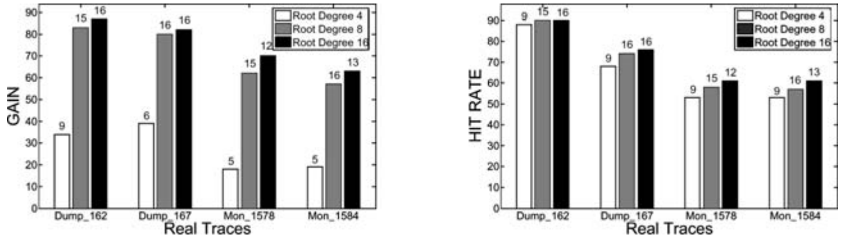


Fig. 10. NoS gain and Hitrate comparison using real traces: root of variable degree

root degrees of 8 and 16, using both synthetic and real traces, in Fig. 9 and 10. The boost in performance between degree 4 and degree 8 is remarkable (120 times the gain of a LRU cache). The higher degree of the root provides more nodes at the top of the tree, with more starting nodes, and enables capturing more traffic at a higher tree-level, increasing the gain. The same reasoning does not apply to hit-rate and the results show little variation from root degree 4.

4 Possible Implementation and Applications

A possible architecture of a NoS implementation is shown in Fig. 11. The data plane performs the per-packet operations, whereas the control plane performs

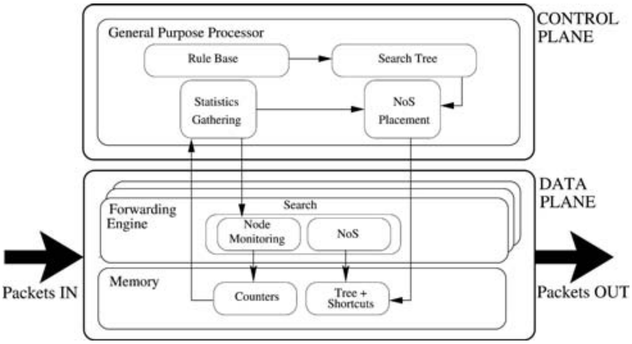
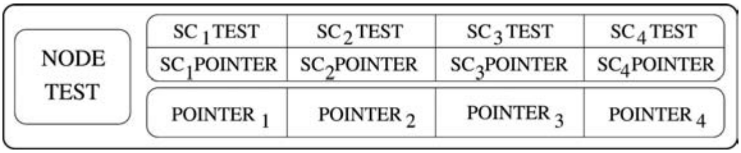


Fig. 11. Relations between building blocks and system resources. The arrows represent the direction of the information exchange. Information is pulled by the statistics gathering engine from the node hit counters and pushed to the search method by the NoS algorithm (shortcuts placement)



NODE TEST	SC_1 TEST	SC_2 TEST	SC_3 TEST	SC_4 TEST
	SC_1POINTER	SC_2POINTER	SC_3POINTER	SC_4POINTER
	POINTER $_1$	POINTER $_2$	POINTER $_3$	POINTER $_4$

Fig. 12. Shortcut starting node possible memory layout. The entry stores the original tree node test and child pointers and the shortcut compressed tests and target pointers

the more complex but less frequent tasks. The data plane consists of one or more forwarding engines and a fast memory (e.g. SRAM), while the control plane is typically implemented on a general purpose processor. An example of such a system is the Intel® IXP^{TM} Network Processor [20].

In the *data plane*, the actual per-packet tree search, including taking the shortcuts, and the hit statistics collection are performed. The hit counters and the shortcut-augmented search-tree reside in fast memory. The actual shortcut entry in the search tree requires to change the memory layout of the original data structure (see Fig. 12). The *control plane* processes include the creation and update of the actual search tree (independent of NoS), the periodic hit counter read and the periodic re-computation of the NoS shortcut placement. The NoS results are periodically uploaded to the search method, updating the shortcuts (but not the search tree itself). The period TA over which the update is performed is set based on the traffic profile and system characteristics.

One possible NoS method application is a denial of service (DoS) attack prevention. Where a cache can be flooded by random independent entries and thus practically disabled, the aggregation of entries into tree nodes would allow the shortcuts to capture such malicious traffic. On the other hand, malicious attacks like the RoQ (Reduction of Quality) attacks [21] against adaptive methods like

NoS can be prevented by placing a second control loop around the adaptation period TA, adjusting it to the rate of changes in the current patterns, and by introducing randomness into TA selection.

5 Conclusions

We have proposed an adaptive data structure to improve the performance of tree-based search methods, using shortcuts along frequently searched paths. The NoS method has been validated through simulation and the results show that NoS outperforms a LRU cache of comparable size in terms of hit rate and memory access reduction. NoS has shown scalable performance even with high number of concurrent flows (70,000+), as the method does not rely on per-flow information, but rather works with the data structure itself.

Despite the good performance when the number of shortcuts is small, it is an open issue how to design a scalable placement strategy, i.e. one that would increase the overall gain when increasing the number of shortcuts. Likewise, efficient update of the shortcut placement remains an open problem. As for the search database's update, there are generally few changes in the search tree structures. If the NoS method was made aware of the changes, the placement could be adjusted, instead of entirely invalidated as in the cache case.

In summary, the shortcut approach offers a viable mechanism for dynamic tree data structure optimization and represents a step towards a fully autonomous packet processing system.

Acknowledgment

We thank Andrew Moore from the Cambridge University Computer Laboratory for providing the networking traces and for very helpful comments.

References

1. R. Kokku et al., *A Case for Run-Time Adaptation in Packet Processing Systems*, In the 2nd Workshop on Hot Topics in Networks (HOTNETS-II), November 2003.
2. M. Ruiz-Sanchez, E. Biersack, and W. Dabbous, *Survey and Taxonomy of IP Address Lookup Algorithms*, IEEE Network Magazine, March 2001.
3. S. Singh, F. Baboescu, G. Varghese, and J. Wang, *Packet Classification Using Multidimensional Cutting*, Proceeding of SIGCOMM, Karlsruhe, Germany, 2003.
4. P. Gupta and N. McKeown, *Packet Classification using Hierarchical Intelligent Cuttings*, Proceedings of Hot Interconnects, 1999.
5. M. Kounavis, A. Kumar, R. Yavatkar, and H. Vin, *Two Stage Packet Classification Using Most Specific Filter Matching and Transport Level Sharing*, Technical Report, Intel Research and Development, September 2004, in submission.
6. V. Snirivasan, S. Suri, and G. Varghese, *Packet Classification using Tuple Space Search*, Proceedings of SIGCOMM, September 1999.
7. T. Lakshman and D. Stiliadis, *High Speed Policy-based Packet Forwarding using Efficient Multi-Dimensional Range Matching*, Proceedings of SIGCOMM, 1998.

8. Y. Zhang, L. Breslau, V. Paxson and S. Shenker, *On the characteristics and origins of internet flow rates*, Proceedings of SIGCOMM, 2002.
9. K. Papagiannaki, N. Taft and C. Diot, *Impact of Flow Dynamics on Traffic Engineering Design Principles*, Proceedings of INFOCOM, 2004.
10. J. Jo, Y. Kim, H. J. Chao and F. Merat, *Internet Traffic Load Balancing using Dynamic Hashing with Flow Volume*, Proceedings of SPIE ITCom, July 2002.
11. K. Claffy, *Internet Traffic Characterization*, Ph.D. Thesis, 1994.
12. S. Ramabhadran and G. Varghese, *Efficient Implementation of a Statistics Counter Architecture*, Proceedings of SIGMETRICS, June 2003.
13. N. Kammenhuber and L. Kencl, *Efficient Statistics Gathering from Tree-Search Methods in Packet Processing Systems*, in press, May 2005.
14. D.D. Sleator and R.E. Tarjan, *Self-Adjusting Binary Search Trees*, Journal of the ACM, July 1985.
15. C. Martínez and S. Roura, *Randomized Binary Search Trees*, Journal of the ACM, March 1998.
16. R. Jain, *Characteristic of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes*, June 1990.
17. C. Partridge et al., *A 50-Gb/s IP Router*, IEEE/ACM Transactions on Networking, 1998.
18. David A. Huffman, *A Method for The Construction of Minimum Redundancy Codes*, Proceedings of IRE, September 1952.
19. A. Moore, J. Hall, C. Kreibich, E. Harris and I. Pratt, *Architecture of a Network Monitor*, Passive and Active Measurement Workshop (PAM), La Jolla, CA, 2003.
20. E.J. Johnson and A.R. Kunze, *IXP2400/2800 Programming*, Intel Press, 2003.
21. M. Guirguis, A. Bestavros and I. Matta, *Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources*, 12th IEEE International Conference on Network Protocols (ICNP), Berlin, Germany, 2004.

Multipath Routing Algorithms for Congestion Minimization

Ron Banner and Ariel Orda

Department of Electrical Engineering,
Technion – Israel Institute of Technology, Haifa 32000, Israel
banner@tx.technion.ac.il
ariel@ce.technion.ac.il

Abstract. Unlike traditional routing schemes that route all traffic along a single path, multipath routing strategies split the traffic among several paths in order to ease congestion. It has been widely recognized that multipath routing can be fundamentally more efficient than the traditional approach of routing along single paths. Yet, in contrast to the single-path routing approach, most studies in the context of multipath routing focused on heuristic methods. We demonstrate the significant advantage of optimal solutions. Hence, we investigate multipath routing adopting a rigorous (theoretical) approach. We formalize problems that incorporate two major requirements of multipath routing. Then, we establish the intractability of these problems in terms of computational complexity. Accordingly, we establish efficient solutions with proven performance guarantees.

Keywords: Routing, Congestion, Algorithms, Optimization, Combinatorics.

1 Introduction

Current routing schemes typically focus on discovering a single "optimal" path for routing, according to some desired metric. Accordingly, traffic is always routed over a single path, which often results in substantial waste of network resources. *Multipath Routing* is an alternative approach that distributes the traffic among several "good" paths instead of routing all traffic along a single "best" path.

Multipath routing can be fundamentally more efficient than the currently used single-path routing protocols. It can significantly reduce congestion in "hot spots", by deviating traffic to unused network resources, thus improving network utilization and providing load balancing [13]. Moreover, congested links usually result in poor performance and high variance. For such circumstances, multipath routing can offer steady and smooth data streams [6].

Previous studies and proposals on multipath routing have focused on *heuristic methods*. In [16], a multipath routing scheme, termed Equal Cost MultiPath (ECMP), has been proposed for balancing the load along multiple shortest paths using a simple round-robin distribution. By limiting itself to shortest paths, ECMP considerably

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 536 – 548, 2005.

reduces the load-balancing capabilities of multipath routing; moreover the *equal* partition of flows along the (shortest) paths (resulting from the round robin distribution) further limits the ability to decrease congestion through load balancing. OSPF-OMP [21] allows splitting traffic among paths *unevenly*; however, the traffic distribution mechanism is based on a heuristic scheme that often results in an inefficient flow distribution. Both [22] and [24] considered multipath routing as an optimization problem with an objective function that minimizes the congestion of the most utilized link in the network; however, they focused on heuristics and did not consider the quality of the selected paths. In [17], a scheme is presented to proportionally split traffic among several "widest" paths that are disjoint with respect to the *bottleneck links*. However, here too, the scheme is heuristic and evaluated by way of simulations.

Simulation results clearly indicate that multipath solutions obtained by *optimal* congestion reduction schemes are *fundamentally* more efficient than the solutions obtained by heuristics. For example, in Section 5, we show that if the traffic distribution mechanism in the ECMP scheme had been optimal, the network congestion would have decreased by more than *three times*; moreover, if paths other than shortest had been allowed, the optimal partition would have decreased the network congestion by more than *ten times*. Hence, the full potential of multipath routing is far from having been exploited.

Accordingly, in this study we investigate multipath routing adopting a rigorous approach, and formulate it as an optimization problem of minimizing network congestion. Under this framework, we consider two fundamental requirements. First, each of the chosen paths should usually be of satisfactory "quality". Indeed, while better load balancing is achieved by allowing the employment of paths other than shortest, paths that are substantially inferior (i.e., "longer") may be prohibited. Therefore, we consider the problem of congestion minimization through multipath routing subject to a restriction on the "quality" (i.e., length) of the chosen paths.

Another practical restriction is on the *number of routing paths per destination*, which is due to several reasons [17]: first, establishing, maintaining and tearing down paths pose considerable overhead; second, the complexity of a scheme that distributes traffic among multiple paths considerably increases with the number of paths; third, often there is a limit on the number of explicitly routing paths (such as label-switched paths in MPLS [19]) that can be set up between a pair of nodes. Therefore, in practice, it is desirable to use as few paths as possible while at the same time minimize the network congestion.

Our Results: Consider first the problem of minimizing the congestion under the requirement to route traffic along paths of "satisfactory" quality. We first show that the considered problem is NP-hard, yet admits a pseudo-polynomial solution. Accordingly, we design two algorithms. The first is an optimal algorithm with a pseudo-polynomial running time, and the second approximates the optimal solution to any desired degree of precision at the (proportional) cost of increasing its running time (i.e., an ε-optimal approximation scheme). In addition, we show that these algorithms can be extended to offer solutions to reliability-related problems.

Consider now the requirement of limiting the number of paths per destination. We show that minimizing the congestion under this restriction is NP-hard as well. Accordingly, we establish a computationally efficient 2-approximation scheme[1]. Then, we generalize the 2-approximation scheme into a bicriteria result and establish a (1+1/r)-approximation scheme that, for any given $r \geq 1$, violates the constraint on the number of routing paths by a factor of at most r. Finally, we broaden the scope of this problem and establish an efficient approximation scheme for the dual problem, which restricts the level of congestion while minimizing the number of paths per destination.

Due to space limits, several proofs and technical details are omitted from this version and can be found (online) in [4].

2 Model and Problem Formulation

A *network* is represented by a directed graph $G(V,E)$, where V is the set of nodes and E is the set of links. Let $N=|V|$ and $M=|E|$. A *path* is a finite sequence of nodes $p=(v_0,v_1,\ldots v_h)$, such that, for $0 \leq n \leq h-1$, $(v_n,v_{n+1}) \in E$. A path is *simple* if all its nodes are distinct. A *cycle* is a path $p=(v_0,v_1,\ldots,v_h)$ together with the link $(v_h,v_0) \in E$ i.e., $(v_0,v_1,\ldots,v_h,v_0)$.[2]

Given a source node $s \in V$ and a target node $t \in V$, $P^{(s,t)}$ is the set of (all) directed paths in $G(V,E)$ from s to t. For each path $p \in P^{(s,t)}$ and link $e \in E$, let $\Delta_e(p)$ count the number of occurrences of e in p. For example, given a non-simple path $p=(v_0,v_1,v_2,v_3,v_1,v_2,v_4)$ and a link $e=(v_1,v_2)$, we have $\Delta_e(p)=2$.

Each link $e \in E$ is assigned a *length* $l_e \in \mathbb{Z}^+$ and a *capacity* $c_e \in \mathbb{Z}^+$. We consider a *link state* routing environment, where each source node has an image of the entire network.

Definition 1: Given a (non-empty) path p, the *length* $L(p)$ of p is defined as the sum of lengths of its links, namely, $L(p) \triangleq \sum_{e \in p} l_e$.

Definition 2: Given a (non-empty) path p, the *capacity* $C(p)$ of p is defined as the capacity of its bottleneck link, namely, $C(p) \triangleq \underset{e \in p}{Min}\{c_e\}$.

Definition 3: Given are a network $G(V,E)$, two nodes $s,t \in V$ and a demand γ. A *path flow* is a real-valued function $f{:}P^{(s,t)} \rightarrow \mathrm{R}^+ \cup \{0\}$ that satisfies the flow demand requirements, i.e., $\sum_{p \in P^{(s,t)}} f_p = \gamma$.

[1] i.e., an algorithm that provides a solution that, in terms of congestion, is within a factor of at most 2 away from the optimum.

[2] As shall be shown, all our solutions consist of *simple* paths exclusively. Cycles and non-simple paths are included in our terminology to simplify the presentation of the solution approach.

Definition 4: Given is a path flow $f{:}P^{(s,t)}\rightarrow \mathrm{R}^{+}\cup\{0\}$ over a network $G(V,E)$. A *link flow* is a real-valued function $f{:}E\rightarrow \mathrm{R}^{+}\cup\{0\}$ that satisfies, for each link $e\in E$, $f_e \triangleq \sum_{p\in P^{(s,t)}} \Delta_e(p)\cdot f_p$.

Definition 5: Given a network $G(V,E)$ and a link flow $\{f_e\}$, the value $\frac{f_e}{c_e}$ is the *link congestion factor* and the value $\max_{e\in E}\left\{\frac{f_e}{c_e}\right\}$ is the *network congestion factor*.

As noted in [3],[13],[22] the network congestion factor provides a good indication of congestion. In [4], we show that the problem of minimizing the network congestion factor is equivalent to the well-known Maximum Flow Problem [1]. Hence, when there are no restrictions on the paths (in terms of the number of paths or the length of each path), one can find a path flow that minimizes the network congestion factor in polynomial time through a standard max-flow algorithm.

We are ready to formulate the two problems considered in this study. The first problem aims at minimizing the network congestion factor subject to a restriction on the "quality" (i.e., length) of each of the chosen paths.

Problem RMP (Restricted Multipath) Given are a network $G(V,E)$, two nodes $s,t\in V$, a length $l_e>0$ and a capacity $c_e>0$ for each link $e\in E$, a demand $\gamma>0$ and a *length restriction* L for each routing path. Find a path flow that minimizes the network congestion factor such that, if $P\subseteq P^{(s,t)}$ is the set of paths in $P^{(s,t)}$ that are assigned a positive flow, then, for each $p\in P$, it holds that $L(p)\leq L$.

Remark 1: For convenience, and without loss of generality, we assume that the length l_e of each link $e\in E$ is not larger than the length restriction L. Clearly, links that are longer than L can be erased.

The next problem considers the requirement to limit the number of different paths over which a given demand is shipped while at the same time minimizing the network congestion factor.

Problem KPR (K-Path Routing) Given are a network $G(V,E)$, two nodes $s,t\in V$, a capacity $c_e>0$ for each link $e\in E$, a demand $\gamma>0$ and a restriction on the number of routing paths K. Find a path flow that minimizes the network congestion factor, such that, if $P\subseteq P^{(s,t)}$ is the set of paths in $P^{(s,t)}$ that are assigned a positive flow, then $|P|\leq K$.
Remark 2: In both problems, the source-destination pair (s,t) is assumed to be connected i.e., $\left|P^{(s,t)}\right|\geq 1$.

3 Minimizing Congestion Under Path Quality Constraints

In this section we investigate Problem RMP, i.e., the problem of minimizing congestion under path quality constraints. We begin by establishing its intractability.

Theorem 1: Problem RMP is NP-hard.

The proof [4] is based on a reduction to the *Partition Problem* [11].

3.1 Pseudo-Polynomial Algorithm for Problem RMP

The first step towards obtaining a solution to Problem RMP is to define it as a linear program. To that end, we need some additional notation.

Recall that we are given a network $G(V,E)$, two nodes $s,t\in V$, a length $l_e>0$ and a capacity $c_e>0$ for each link $e\in E$, a demand $\gamma>0$ and a length restriction L for each routing path. Let α be the network congestion factor. Denote by f_e^{λ} the total flow along $e=(u,v)\in E$ that has been routed from s to u through paths with a total length of λ. Finally, for each $v\in V$, denote by $O(v)$ the set of links that emanate from v, and by $I(v)$ the set of links that enter that node, namely $O(v)=\{(v,l)/(v,l)\in E\}$ and $I(v)=\{(w,v)/(w,v)\in E\}$. Then, Problem RMP can be formulated as a linear program over the variables $\left\{\left\{f_e^{\lambda}\right\},\alpha\right\}$, as specified in Fig 1.

Program RMP $\left(G(V,E),\{s,t\},\{l_e\},\{c_e\},\gamma,L\right)$

Minimize α (1)

Subject to:

$$\sum_{e\in O(v)} f_e^{\lambda} - \sum_{e\in I(v)} f_e^{\lambda-l_e} = 0 \qquad \forall v\in V\setminus\{s,t\},\forall\lambda\in[0,L] \tag{2}$$

$$\sum_{e\in O(s)} f_e^{\lambda} - \sum_{e\in I(s)} f_e^{\lambda-l_e} = 0 \qquad \forall\lambda\in[1,L] \tag{3}$$

$$\sum_{e\in O(s)} f_e^{0} = \gamma \tag{4}$$

$$\sum_{\lambda=0}^{L} f_e^{\lambda} \le c_e\cdot\alpha \qquad \forall e\in E \tag{5}$$

$$f_e^{\lambda} = 0 \qquad \forall e\in E,\ \lambda\notin[0,L-l_e] \tag{6}$$

$$f_e^{\lambda} \ge 0 \qquad \forall e\in E,\ \lambda\in[0,L] \tag{7}$$

$$\alpha\ge 0 \tag{8}$$

Fig. 1. Program RMP

The objective function (1) minimizes the network congestion factor. Constraints (2), (3) and (4) are nodal flow conservation constraints. Equation (2) states that the traffic flowing out of node v, which has traversed through paths $p\in P^{(s,v)}$ of length $L(p)=\lambda$, has to be equal to the traffic flowing into node v, through paths $p'\in P^{(s,u)}$ and links e=(u,v)∈E, such that $L(p')+l_e=\lambda$; since $\lambda\in[0,L]$, the length restriction is obeyed; finally, equation (2) must be satisfied for each node other than the source s and the target t. Equation (3) extends the validity of equation (2) to hold for traffic that encounters source s after it has already passed through paths with non-zero length. Informally, equation (3) states that "old" traffic that emanates from s *not* for the first time (through a directed cycle that contains the source s) must satisfy the nodal flow conservation constraint of equation (2), which *solely* focuses on nodes from $V\setminus\{s,t\}$. Equation (4) states that the total traffic flowing out of source s, which has traversed paths of length $L=0$, must be equal to the demand γ. Informally, equation (4) states that the total "new" traffic that emanates from the source s for the first time must satisfy the flow demand γ. Equation (5) is the link capacity utilization constraint. It states that the maximum link utilization is not larger than the value of the variable α i.e., the network congestion factor is at most α. Expression (6) rules out non-feasible flows and Expressions (7) and (8) restrict all variables to be non-negative.

We can solve Program RMP (Fig. 1) using any polynomial time algorithm for linear programming [15]. The solution to the problem is then achieved by decomposing the output of Program RMP (i.e., link flow $\{f_e^\lambda\}$) into a path flow that satisfies the length restriction L. This is done by modifying the *flow decomposition algorithm* [1] (that transforms link flows $\{f_e\}$ into path flows $\{f_p\}$) in order to consider length restrictions i.e., transform link flows with "lengths" $\{f_e^\lambda\}$ into path flows that obey the length restrictions. Due to space limits, the description of this algorithm is omitted and can be found in [4].

In the remainder of this subsection we consider the complexity of the overall solution (henceforth, Algorithm RMP), which is dominated by the complexity of Program RMP [4]. It follows from [15] that the complexity incurred by solving Program RMP is polynomial both in the number of variables $\{f_e^\lambda\}$ and in the number of constraints needed to formulate the linear program. Thus, since both of these numbers are in the order of $M\cdot L$, the complexity of Algorithm RMP is polynomial in $O(M\cdot L)$ i.e., Algorithm RMP is a *pseudo-polynomial algorithm* [11]. Thus, whenever the value of L is polynomial in the size of the network, Algorithm RMP is a *polynomial optimal* algorithm for Problem RMP. One such case is when the *hop count* metric is considered (i.e., $l_e\equiv 1$), since then $L\leq N-1$.

3.2 ε-Optimal Approximation Scheme for Problem RMP

In the previous subsection we established an optimal polynomial solution to Problem RMP for the case where the length restrictions are sufficiently small. In this subsection we turn to consider the solution to Problem RMP for *arbitrary* length restrictions. As Theorem 1 establishes that Problem RMP is NP-hard for this general case, we focus on the design of an efficient algorithm that approximates the optimal solution.

Our main result in this setting is the establishment of an ε–optimal approximation scheme, which is termed the *RMP Approximation Scheme*. This scheme is based on Algorithm RMP, specified in the previous subsection, which was shown to have a complexity that is polynomial in $M\cdot L$. Given an instance of Problem RMP and an approximation parameter ε, we reduce the complexity of Algorithm RMP by first scaling down the length restriction L by the factor $\Delta \triangleq \frac{L\cdot\varepsilon}{N}$ and then rounding it into an integer value. Obviously, as a result, we must also scale down the length of each link. However, in order to ensure that the optimal network congestion factor does not increase, we relax the constraints of the new instance with respect to the constraints of the original instance. Specifically, after we scale down the length restriction and the length of each link by the factor Δ, we round *up* the length restriction and round *down* the length of each link. Then, we invoke Algorithm RMP over the new instance, in order to construct a path flow that minimizes congestion while satisfying the relaxed length restrictions. Finally, we convert each non-simple path in the output of Algorithm RMP into a simple path by eliminating loops; this is essential, since the total error in the evaluation of the length of each path depends on the hop count. In Theorem 2, we establish that the resulting path flow violates the length restriction by a factor of at most $(1+\varepsilon)$ and has a network congestion factor that is not larger than the optimal network congestion factor. The proof can be found in [4].

Theorem 2: Given an instance $<G,\{s,t\},\{c_e\},\{l_e\},\gamma,L>$ of problem RMP and an approximation parameter ε, the *RMP Approximation Scheme* has a complexity that is polynomial in $1/\varepsilon$ and the size of the network; moreover, the output of the scheme is a path flow f that satisfies the following:

a. $\sum_{p\in P^{(s,t)}} f_p = \gamma$ i.e., the flow demand requirement is satisfied.

b. If α^* is the network congestion factor of the optimal solution, then, for each e$\in E$, it holds that $\sum_{p\in P^{(s,t)}} \Delta_e(p)\cdot f_p \le \alpha^*\cdot c_e$, i.e., the network congestion factor is at most α^*.

c. For each path $p\in P^{(s,t)}$, if $f_p>0$ then p is *simple* and $L(p)\le(1+\varepsilon)\cdot L$, i.e., the length restriction is violated by a factor of at most $(1+\varepsilon)$.

3.3 Further Results

In the following, we outline two important extensions to Problem RMP.

Multi-commodity Extensions: In [4], we consider a multi-commodity extension of Problem RMP, i.e., a problem with several source-destination pairs. Following basically the same lines as in Subsections 3.1 and 3.2, we present a pseudo-polynomial solution for this problem and establish an ε-optimal approximation.

End-to-End Reliability Constraints: In [4], we also consider the increased vulnerability to failures when multipath routing is employed in order to balance the network load. Indeed, when traffic is split among multiple paths, a failure in each routing path may result in the failure of the entire transmission. In [4] we formulate the problem and show that it is computationally intractable. However, we show there that the RMP Approximation Scheme can be modified in order to constitute an ε–optimal approximation scheme for the reliability problem.

4 Minimizing Congestion with K Routing Paths

In this section we solve Problem KPR, which minimizes congestion while routing traffic along at most K different paths. In [4], we show that Problem KPR admits a (straightforward) polynomial solution when the restriction on the number of paths is larger than the number of links M (i.e., $K \geq M$). However, we show in [4] that, in the more interesting case where $K<M$, the problem is NP-hard. Accordingly, in this section we present a 2-approximation scheme for $K<M$.

Our approximation scheme is based on solving an auxiliary problem that minimizes congestion while restricting the flow along each path to be *integral* in γ/K. In order to formulate the corresponding problem, consider first the following definition.

Definition 7: Given are a network $G(V,E)$, a capacity $c_e>0$ for each link $e \in E$, a demand γ and an integer K. A path flow $f{:}P \rightarrow \mathbb{R}^+ \cup \{0\}$ is said to be γ/K-*integral*, if for each path $p \in P^{(s,t)}$, it holds that f_p is a multiple of γ/K.

Problem Integral Routing: Given are a network $G(V,E)$, two nodes $s,t \in V$, a capacity $c_e>0$ for each link $e \in E$, a demand $\gamma>0$ and an integer K. Find a γ/K-integral path flow that minimizes the network congestion factor, such that the demand γ is satisfied.

4.1 Solving the Integral Routing Problem

The following observation shall be used in order to construct a polynomial solution to the Integral Routing Problem. The proof can be found in [4].

Lemma 1: Given an instance $<G,\{s,t\},\{c_e\},\gamma,K>$ of the Integral Routing Problem, the optimal network congestion factor is included in the set $\bar{\alpha} \triangleq \left\{ \frac{i \cdot \gamma}{K \cdot c_e} \middle| e \in E, i \in [0,K] \cap \mathbb{Z} \right\}$[1].

We now introduce *Procedure Test*, which is given an instance $<G,\{s,t\},\{c_e\},\gamma,K>$ of the Integral Routing Problem and a restriction α on the network congestion factor. Procedure Test performs three sequential steps. Initially, it multiplies all link capacities by a factor of α in order to impose the restriction on the network congestion factor; indeed, multiplying all capacities by α assures that the flow f_e along each link

[1] Observe that the size of $\bar{\alpha}$ is polynomial in the network size, namely: $|\bar{\alpha}| \leq M \cdot (K+1) = O(M^2)$.

$e \in E$ is at most $\alpha \cdot c_e$; therefore, for each $e \in E$, the link congestion factor f_e/c_e, and, in particular, the network congestion factor $\max_{e\in E}\{f_e/c_e\}$, are at most α. Next, the procedure rounds down the capacity of each link to the nearest multiple of γ/K; since the flow over each path in every solution to the Integral Routing Problem is γ/K-integral, such a rounding has no effect on the capability to transfer the flow demand γ. Finally, the procedure applies any standard maximum flow algorithm that returns an integral link flow when all capacities are integral. Since all capacities are γ/K-integral, the maximum flow algorithm determines a γ/K-integral link flow that transfers the maximum amount of flow without violating the restriction α on the network congestion factor. If this link flow succeeds in transferring at least γ flow units from s to t, then the procedure returns it. Otherwise, the procedure fails.

Theorem 3: Given is an instance $<G,\{s,t\},\{c_e\},\gamma,K>$ of the Integral Routing Problem. Denote by α^* the corresponding optimal network congestion factor. Then, Procedure Test succeeds for the input $<G,\{s,t\},\{c_e\},\gamma,K,\alpha>$ *iff* $\alpha \geq \alpha^*$.
The proof appears in [4].

Theorem 3 has two important implications that enable to construct an efficient solution to the Integral Routing Problem. First, the theorem establishes that the smallest α for which Procedure Test succeeds with the input $<G,\{s,t\},\{c_e\},\gamma,K,\alpha>$ is equal to α^*. Therefore, if S is a finite set that includes the optimal network congestion factor α^* and α is the smallest network congestion factor in S such that Procedure Test succeeds for the input $<G,\{s,t\},\{c_e\},\gamma,K,\alpha>$, then $\alpha=\alpha^*$. This fact, together with the fact that the set $\overline{\alpha}$ includes α^* (as per Lemma 1), imply that, for every instance $<G,\{s,t\},\{c_e\},\gamma,K>$ of Problem Integral Routing, the optimal network congestion factor α^* is the smallest $\alpha \in \overline{\alpha}$ such that Procedure Test succeeds for the input $<G,\{s,t\},\{c_e\},\gamma,K,\alpha>$. Moreover, since in case of a success Procedure Test returns the corresponding link flow, finding the smallest $\alpha \in \overline{\alpha}$ such that Procedure Test succeeds identifies a link flow with a network congestion factor of at most α^*.

The second implication of Theorem 3 enables to employ a *binary search* when we seek the smallest $\alpha \in \overline{\alpha}$ such that Procedure Test succeeds. Indeed, it follows from Theorem 3 that, when Procedure Test succeeds for $\alpha_1 \in \overline{\alpha}$, it succeeds for all $\alpha \in \overline{\alpha}$, $\alpha \geq \alpha_1$; and when it fails for $\alpha_2 \in \overline{\alpha}$, it fails for all $\alpha \in \overline{\alpha}$, $\alpha \leq \alpha_2$; thus, if Procedure Test succeeds for $\alpha_1 \in \overline{\alpha}$ (alternatively, fails for $\alpha_2 \in \overline{\alpha}$) it is possible to eliminate from further consideration all the elements of $\overline{\alpha}$ that are larger than α_1 (correspondingly, smaller than α_2).

Remark 3: Note that performing a binary search over $\overline{\alpha}$ requires *sorting* all the elements of $\overline{\alpha}$, which consumes $O\left(\left|\overline{\alpha}\right| \cdot \log\left|\overline{\alpha}\right|\right) \leq O\left(M^2 \cdot \log N\right)$ operations [10].

Thus, we conclude that the employment of a binary search so as to find the smallest $\alpha \in \overline{\alpha}$ for which Procedure Test succeeds, establishes a link flow that has the minimal network congestion factor. The optimal solution is then achieved by decomposing the resulting link flow into a path flow via the flow decomposition algorithm [1]. Due to space limits, the formal description of this algorithm, termed *Algorithm Integral Routing*, is omitted and can be found in [4]. Our discussion is summarized by the following theorem, which establishes that Algorithm Integral Routing solves Problem Integral Routing. Its proof appears in [4].

Theorem 4: Given is an instance $<G,\{s,t\},\{c_e\},\gamma,K>$ of Problem Integral Routing. If Algorithm Integral Routing returns Fail, then there is no feasible solution for the given instance; otherwise, the algorithm returns a γ/K-integral path flow that transfers at least γ flow units from s to t along simple paths, such that the network congestion factor is minimized.

Remark 4: It is easy to show [4] that the computational complexity of Algorithm Integral Routing is $O(M{\cdot}logN{\cdot}(M+N{\cdot}logN))$.

4.2 A 2-Approximation Scheme for Problem KPR

Finally, we are ready to establish a solution for Problem KPR. To that end, we show that the solution of the Integral Routing Problem can be used in order to establish a *constant approximation scheme* for Problem KPR. The approximation scheme is based on the following key observation, which links between the optimal solution of Problem Integral Routing and the optimal solution of Problem KPR.

Theorem 5: Given are a network $G(V,E)$ and a demand of γ flow units that has to be routed from s to t. If f_1 is a γ/K-integral path flow that has the minimum network congestion factor and f_2 is a path flow that minimizes its network congestion factor while routing along at most K paths, then the network congestion factor of f_1 is at most twice the network congestion factor of f_2.

Proof: Suppose that f_1 and f_2 satisfy the assumptions of the Theorem. Let α_1 and α_2 denote the network congestion factor of path flows f_1 and f_2, respectively. We have to show that $\alpha_1 \leq 2{\cdot}\alpha_2$.

Out of the path flow f_2, we construct a γ/K-integral path flow that ships at least γ flow units from s to t and has a network congestion factor of at most $2{\cdot}\alpha_2$. Clearly, such a construction implies that the network congestion factor of every *optimal* γ/K-integral path flow that ships γ flow units from s to t is at most $2{\cdot}\alpha_2$; in particular, since f_1 is one such optimal γ/K-integral path flow, such a construction establishes that $\alpha_1 \leq 2{\cdot}\alpha_2$.

With this goal in mind, define the following construction. First, double the flow along each routing path that f_2 employs; obviously, the resulting path flow transfers $2{\cdot}\gamma$ flow units from s to t along at most K routing paths while yielding a network congestion factor of $2{\cdot}\alpha_2$. Then, *round down* the (doubled) flow along each routing path to the nearest multiple of γ/K; in this process, the flow along each path is reduced by at most γ/K flow units. Hence, since there are no more than K routing paths, the total flow from s to t is reduced by at most γ units; therefore, since before the rounding operation exactly $2\cdot\gamma$ flow units were shipped from s to t, it follows that after rounding is performed, the resulting path flow transfers at least γ flow units from s to t.

Thus, we have identified a γ/K-integral path flow that transfers at least γ flow units from s to t. In addition, since prior to the rounding operation the network congestion factor is $2\cdot\alpha_2$ and the rounding can only reduce flow, the network congestion factor of the constructed path flow is at most $2\cdot\alpha_2$. ■

Note that, given a network $G(V,E)$ and a demand γ that needs to be routed over at most K paths, *every* γ/K-integral path flow satisfies the requirement to ship the demand γ on at most K different paths. On the other hand, it has been established in Theorem 5 that the network congestion factor obtained by an optimal γ/K-integral path flow is at most twice the network congestion factor of an optimal flow that admits at most K routing paths. Thus, computing a γ/K-integral path flow that has the minimum network congestion factor satisfies the restriction on the number of routing paths and obtains a network congestion factor that is at most twice larger than the optimum. We summarize the above discussion in the following corollary, which yields an approximation scheme for Problem KPR.

Corollary 1: Given are a network $G(V,E)$, a demand γ and a restriction on the number of routing paths K. The employment of Algorithm Integral Routing for the establishment of a γ/K-integral path flow that minimizes the network congestion factor provides a 2-approximation scheme for Problem KPR with a complexity of $O(M\cdot logN\cdot(M+ N\cdot logN))$.

4.3 Further Results

In [4], we generalize the result of this section into a *bicriteria* result. Specifically, for any given $r\geq1$, we establish a $(1+1/r)$-approximation scheme that violates the constraint on the number of paths by a factor of at most r. Note that, for $r = 1$, the corresponding scheme obtains the same performance guarantees as in Subsection 4.2 above. In addition, in [4] we consider the *dual* problem, which restricts the network congestion factor while minimizing the number of routing paths, and present a corresponding approximation scheme.

5 Simulation Results

In this section, we present a comparison between an optimal solution to multipath routing and that provided by a heuristic scheme such as the (popular) Equal Cost MultiPath (ECMP) routing scheme.

We generated 10,000 random topologies, following the lines of [23][1]. For each topology, we conducted the following measurements: (a) we measured the network congestion factor produced by invoking ECMP; (b) we measured the network

[1] Due to space limits, we omit the details of this construction, which can be found in [4].

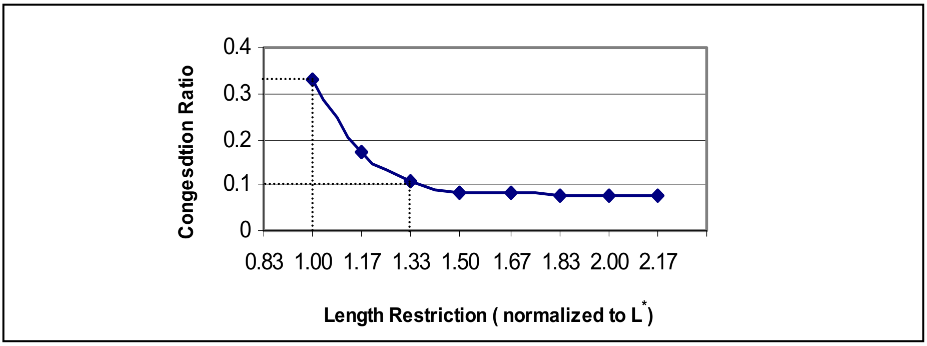


Fig. 2. The ratio between the network congestion produced by an optimal multipath routing assignment (for several length restrictions) and the network congestion produced by ECMP

congestion factor produced by an optimal assignment of traffic to shortest paths and to paths with a length that is equal to 1.17·L^*, 1.33·L^*, 1.5·L^*, 1.67·L^*, 1.83·L^*,2·L^* and 2.17·L^*, where L^* is the length of a shortest path. Our results are summarized in Fig. 2. Note that if the ECMP scheme had an optimal traffic distribution mechanism, the network congestion factor could be reduced by a *factor of 3*. Moreover by relaxing the requirement to route along shortest paths by 33%, the network congestion factor is *10 times smaller* than with the standard ECMP. Thus, by employing Algorithm RMP or its *e*-optimal approximation with $L \approx 1.33 \cdot L^*$, congestion can be reduced by a factor of *10* with respect to that produced by ECMP.

6 Conclusion

Previous multipath routing schemes for congestion avoidance focused on heuristic methods. Yet, our simulations indicate that optimal congestion reduction schemes are *significantly* more efficient. Accordingly, we investigated multipath routing as an optimization problem of minimizing network congestion, and considered two fundamental problems. Although both have been shown to be computationally intractable, they have been found to admit efficient approximation schemes. Indeed, for each problem, we have designed a *polynomial* time algorithm that approximates the optimal solution by a (small) *constant* approximation factor.

While this study has laid the algorithmic foundations for two fundamental multipath routing problems, there are still many challenges to overcome. One major challenge is to establish an efficient unifying scheme that combines the two problems. Furthermore, as in practice there may be a need for simpler solutions, another research challenge is the development of approximations with lower computational complexity. Finally, as discussed in [4], multipath routing offers reach ground for research also in other contexts, such as survivability, recovery, network security and energy efficiency. We are currently working on these issues and have obtained several results regarding survivability [5].

References

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, "Network Flows: Theory, Algorithm, and Applications", Prentice Hall, 1993.

[2] D. Awduche, J. Malcolm, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS" , Internet Draft, April, 1998.

[3] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, "Requirements for Traffic Engineering Over MPLS", IETF RFC 2702, September 1999.

[4] R. Banner and A. Orda, "Multipath Routing Algorithms for Congestion Minimization", CCIT Report No. 429, Department of Electrical Engineering, Technion, Haifa, Isreal, 2004. Available from: http://www.ee.technion.ac.il/people/ron/Congestion.pdf

[5] R. Banner and A. Orda, "The Power of Tuning: a Novel Approach for the Efficient Design of Survivable Networks", In Proc. IEEE ICNP 2004 (**Best Paper Award**).

[6] D. Bertsekas and R. Gallager, “Data networks”, Prentice-Hall, 1992.

[7] Allan Borodin and Ran El-Yaniv, "Online Computation and Competative Analysis", Cambridge University Press, 1998.

[8] I. Cidon, R. Rom and Y. Shavitt, "Analysis of Multi-Path Routing", IEEE Transactions on Networking, v. 7, No. 6, pp. 885–896, 1999.

[9] S. Chen and K. Nahrstedt, “An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions”, IEEE Network, v. 12, No. 6, pp. 64-79, December 1998.

[10] T. Cormen, C. Leiserson, and R. Rivest, Introduction to Algorithms. Cambridge, MA: The MIT Press, 2001.

[11] M. R. Garey and D. S. Johnson, "Computers and Intractability", W.H. Freeman and Co., 1979.

[12] A. V. Goldberg and R. E. Tarjan, "A New Approach to The Maximum Flow Problem", Journal of ACM, v.35, No. 4, pp. 921-940, 1988.

[13] S. Iyer, S. Bhattacharyya, N. Taft, N. McKeoen, C. Diot, “A measurement Based Study of Load Balancing in an IP Backbone”, Sprint ATL Technical Report, TR02-ATL-051027, May 2002.

[14] Y. Jia, Ioanis Nikolaidis and P. Gburzynski, "Multiple Path QoS Routing", Proceedings of ICC'01 Finland, pp. 2583-2587, June 2001.

[15] N. Karmarkar, "A New Polynomial-Time Algorithm For Linear Programming", Combinatorica, vol. 4, pp. 373-395, 1984.

[16] J.Moy, ”OSPF Version 2”, IETF RFC 2328, April 1998.

[17] S. Nelakuditi and Zhi-Li Zhang, “On Selection of Paths for Multipath Routing”, In Proc. IWQoS, Karlsruhe, Germany, 2001.

[18] V. Paxson, "End-to-End Routing Behavior in the Internet", in proc. ACM SIGCOMM, 1996.

[19] E. Rosen, A. Viswanathan, and R. Callon. "Multiprotocol Label Switching Architecture". IETF RFC 3031, 2001.

[20] D. Thaler and C. Hopps, “Multipath Issues in Unicast and Multicast Next-Hop Selection”, IETF RFC 2991, 2000.

[21] C. Villamizar, OSPF Optimised Multipath (OSPF-OMP), Internet Draft, 1998.

[22] Y. Wang and Z. Wang, “Explicit Routing Algorithms For Internet Traffic Engineering”, In Proc. ICCN'99, Boston, October 1999.

[23] B. M. Waxman, *Routing of Multipoint Connections*, IEEE Journal on Selected Areas in Communications, 6:1617-1622,1988.

[24] A. E. I. Widjaja, "Mate: MPLS Adaptive Traffic Engineering", Internet Draft, August, 1998.

Improving TCP in Wireless Networks with an Adaptive Machine-Learnt Classifier of Packet Loss Causes

Ibtissam El Khayat, Pierre Geurts, and Guy Leduc

Department of Electrical Engineering and Computer Science,
University of Liège,
Institut Montefiore - B28 - Sart Tilman,
Liège 4000 - Belgium

Abstract. TCP understands all packet losses as buffer overflows and reacts to such congestions by reducing its rate. In hybrid wired/wireless networks where a non negligible number of packet losses are due to link errors, TCP is unable to sustain a reasonable rate. In this paper, we propose to extend TCP Newreno with a packet loss classifier built by a supervised learning algorithm called 'decision tree boosting'. The learning set of the classifier is a database of 25,000 packet loss events in a thousand of random topologies. Since a limited percentage of wrong classifications of congestions as link errors is allowed to preserve TCP-Friendliness, our protocol computes this constraint dynamically and tunes a parameter of the classifier accordingly to maximise the TCP rate. Our classifier outperforms the Veno and Westwood classifiers by achieving a higher rate in wireless networks while remaining TCP-Friendly.

1 Introduction

TCP has been deployed in the eighties. Its congestion control is based on the fact that packet losses are mainly due to buffer overflows and it works quite well in such situations. However, nowadays, many applications use TCP as their transport protocol and hence pass through wireless links, which become common in the Internet. Over these links, packet losses are not due anymore only to overflows but can also be caused by link errors. TCP, which has no mechanism to distinguish packet loss causes, reduces its rate at each packet loss. This reduction is not justified when there is no congestion and the consequence is that the throughput of TCP over wireless link is lower than what it could be.

To increase its throughput, TCP should avoid reacting to a packet loss due to a link error as it does when it faces a congestion. Two possibilities have been proposed in the literature. The first one is to hide link error losses from the sender (for example by splitting the TCP connection [1] or retransmitting in link layer). These solutions assume however the support of the network. Splitting the TCP connection has another important drawback which is the violation of the principle of end-to-end TCP: It allows to send an acknowledgement to the sender

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 549–560, 2005.

before the sink has received the packet. The second approach, which is the one adopted in this paper, is to endow one of the end systems with an algorithm that classifies the packet loss causes. Following this strategy, Veno [8] and Westwood [15] use some simple test to classify loss causes. Veno estimates the backlog and consider that the loss is due to a congestion if the backlog is higher than 3. Westwood classifies loss causes with a test that compares the current RTT to 1.4.RTT$_{min}$ (where 1.4.RTT$_{min}$ is the smallest RTT estimated by TCP). In our opinion, one such simple test is not sufficient to make a good classification of loss causes in general and indeed, our simulations below will show that these two tests do not classify very well the loss causes in practice. Liu and al. [12] use hidden Markov models based on RTT values to make the discrimination. It has been shown in [2] that there is no correlation between the round-trip-time and the loss cause. Indeed, a modification in the return path affects the round-trip-time without affecting the loss cause.

Therefore, we propose in this paper to infer a more complete model to classify loss causes that combines several indicators instead of one. Characterising analytically the network conditions leading to a certain type of packet loss is difficult because real networks are very complex systems but also because their behaviours depend on a large number of random external factors (user behaviours, current topologies...). On the other hand, it is quite easy to simulate the network behaviour (e.g. with a network simulator like `ns-2`) or to gather data from observation of the behaviour of a real network. This is the typical situation where automatic learning techniques are useful. These algorithms are general techniques to extract a model of a system only from data obtained either by direct observations or by simulation of this system. Of interest for our problem is supervised learning which focuses on the approximation of an input/output relationship only from observations of examples of this relationship.

More specifically, in this paper, we propose to apply a particular learning algorithm called decision tree boosting to automatically design a model for discriminating the two possible packet loss causes and then use this model at best to improve the performance of TCP in wired/wireless networks. The paper is structured as follows. In Section 2, we give a short general introduction to supervised learning. Learning algorithms require the generation of a database from which to infer a model. Section 3 describes how we generate this database and evaluates the performance of decision tree boosting applied to this problem. In Section 4, we explain the design of our improved TCP constructed upon Boosting. Finally, Section 5 evaluates our extension of TCP with several simulations.

2 Supervised Learning

Supervised learning is the part of the field of machine learning which focuses on modelling input/output relationships. More precisely, the goal of supervised learning is to identify a mapping from some input variables to some output variable on the sole basis of a sample of observations of these variables. Formally, the sample of observations is called the learning sample LS and is a set of input/output pairs, $LS = \{< x_1, y_1 >, < x_2, y_2 >, ..., < x_N, y_N >\}$, where x_i is the vector of values of the input variables (also called the attributes) corresponding to the i^{th} observation (also called an object) and y_i is its output value. Attribute values may be discrete or continuous. The goal of supervised learning can be formulated as follows: From a learning sample LS, find a function $f(x)$ of the input attributes that predicts at best the outcome of the output attribute y for any *new unseen* value of x. When the output takes its values in a discrete set $\{C_1, C_2, ..., C_m\}$, we talk about a classification problem and when it is continuous, we talk about a regression problem.

This problem is solved by a (supervised) learning algorithm. Loosely speaking, a learning algorithm receives a learning sample and returns a function f (an hypothesis or a model) which is chosen in a set of candidate functions (the hypothesis space). Among the most popular supervised learning algorithms, there are decision trees and neural networks. In this paper, we will use an algorithm called decision tree boosting, which is a powerful extension of decision tress. For a complete reference on supervised learning algorithms, see for example [11].

3 Loss Classification by Supervised Learning

In this section, we focus on the problem of the derivation and the evaluation of a model for predicting loss causes. The question of the application of this model to improve TCP will be addressed in the next sections.

3.1 The Database

To solve our problem of losses classification, each observation $< x_i, y_i >$ of the learning sample will be an input/output pair where the inputs x_i are some variables that describe the state of the system at the occurrence of a loss and the (discrete) output y_i is either C to denote a loss due to a congestion or LE to denote a loss due to a link error.

To make the model generally applicable, the observations in the database must be as much as possible representative of the conditions under which we will apply the classification model. So, the database generation should take into account all the uncertainties we have a priori about the topology of the networks, the user behaviours, and the protocols. We describe below the way we generated our observations and we discuss our choice of input variables. Our database and the TCL code used to generate our topologies can be found at [4].

Database Generation. The database was generated by simulations with the network simulator `ns-2`. To generate our observations of losses, we have used the following procedure: a network topology is generated randomly and then the network is simulated during a fixed amount of time, again by generating the traffic randomly. At the end of the simulation, all losses that have occurred within this time interval are collected in the database. This procedure is repeated until we have a sufficient number of observations in the database. In practice, the larger the learning sample, the better it is for supervised learning algorithms. In our study, we have collected 35,441 losses that correspond to more than one thousand different random topologies.

To generate a random topology, we first select a random number of nodes (between 10 and 600) and then choose randomly the connections between these nodes. The links are chosen simplex to avoid symmetrical topologies. The bandwidth, the propagation delay and the buffer size of the links were chosen randomly. The bandwidth is chosen between 56Kb/s and 100Mb/s while the propagation delay varies between 0.1ms and 500ms. As Droptail is the most widely deployed policy [5], our simulations all use this latter policy.

The number of wireless links, their place in the topology, the error model and the loss rate were also drawn at random. The error models are either the simple uniform error model, to mimic random losses, or the two-state Gilbert-Elliott model, to mimic bursty losses. Although these two models are often used to simulate wireless losses (e.g., [15], [10]), they do not perfectly match real wireless links. Note however that taking into account more realistic models in our approach is straightforward.

Concerning the traffic, 60% of the flows at least were TCP Newreno flows and the others were chosen randomly among TCP and other types of traffic proposed by `ns-2` and based on UDP. The senders, the receivers, the packets size and the duration of each traffic were set randomly. Thus, the database contains losses belonging to short and long TCP sessions. This random choice of traffic length allows us to avoid making any assumption about the network load which is randomised in the database.

The Choice of the Inputs. At the end system, the information we can measure to predict a congestion is the inter-packet times and the one-way delay. These measures can be obtained at both sides. The one-way delay, computed by one of the two entities, is the difference between the timestamp of the acknowledgement and the timestamp of the TCP packet, and is actually the real one-way delay minus the difference between the clocks of the sender and the receiver. This latter difference is not important in our study since we will see below that our inputs are based only on relative variation of the one-way delay.

To compute our inputs, we use the values of the inter-packet times and the one-way delay for the three packets following the loss[1] and the packet that precedes it. To make the model independent of the absolute values of these measures, we normalised these values in different ways. To this end, we relate them using various functions to the average, the standard deviation, the minimum, and the maximum of the one-way delay and inter-packet time for the packets that are sent during the last two round-trip-times before the loss. In total, this normalisation results in about 40 inputs.

[1] We consider only losses detected by triple duplicates.

3.2 Decision Tree Boosting

There exist many different learning algorithms that could be used for our problem. In [9], we have carried out experiments with several methods, among which artificial neural networks, the k nearest neighbors, and several tree-based algorithms. In this paper, we concentrate our discussion on the method that gives the best results for our problem, which turns out to be decision tree boosting.

Decision tree induction [3] is one of the most popular learning algorithm. A decision tree represents a classification model by a tree where each interior node is labeled with a test on one input attribute and each terminal node is labeled with a value of the output (here C or LE). To classify an observation with such a tree, we simply propagate it from the top node to a terminal node according to the test issues and the prediction for this observation is the class associated with the terminal node.

In supervised learning, ensemble methods are generic techniques that improve a learning algorithm by learning several models (from the same learning sample) and then aggregating their predictions. Boosting [7] is a particular ensemble method where the models are built in sequence. Each model is built by increasing the weights of the learning sample cases that are misclassified by the previous models in the sequence. Then, the prediction of the resulting ensemble of models is the majority class among the classes given by all models. When applied on the top of decision trees, this method often improves very importantly the accuracy with respect to a single tree. Actually, this combination of boosting and decision tree is often considered as one of the best supervised learning algorithm for classification problems [11]. In our experiments, the number of trees constructed by boosting was fixed to 25.

3.3 Model Evaluation

Usually, the error rate of the model at classifying loss causes in the learning sample is very small since the learning algorithm explicitely tries to minimize this error. Thus, this error is not a good indication of the ability of the model at classifying losses in new unseen topologies. To get a more reliable estimate of the error of the classifier, we have thus randomly divided the database into two parts: a learning sample that is used to learn the model and a test sample on which the resulting classifier is tested. Since the losses in both samples are obtained from different topologies, the error rate of the model at classifying losses in the test sample gives a good idea of the probability of misclassification of our model in a new situation.

When evaluating the model on the test sample, there are two errors of interest: the probability that the model misclassifies a congestion as a link error and the probability that it misclassifies a link error as (a loss due to) a congestion. We will denote these errors respectively Err_C and Err_{LE}. Of course, it is important to minimize these two types of errors but we will show later that if one decreases the other increases. So, independently of the application of the model, it is very desirable to be able to favour the accuracy of the prediction of one type of loss over the other.

Actually, besides the class prediction, Boosting also provides an estimate of the probability of each class, C or LE, given the input x, i.e. two numbers $\hat{P}(C|x)$ and $\hat{P}(LE|x)$ such that $\hat{P}(C|x) + \hat{P}(LE|x) = 1$. The class given by the model is then LE if $\hat{P}(LE|x)$ is greater than a threshold P_{th} and C otherwise. By default, the value of P_{th} is fixed to 0.5 so as to treat each class fairly. However, by changing the threshold, we can easily favour the accuracy of the prediction of one class over the other. By taking P_{th} lower than 0.5, more losses will be predicted as due to link error and hence, we will decrease Err_{LE} and increase Err_C. On the opposite, by taking P_{th} greater than 0.5, we will decrease Err_C and increase Err_{LE}. So, this parameter allows us to obtain different classification models with different tradeoffs between the two types of error. It is also important to note that the user can choose the tradeoff that fits his application without re-running the learning algorithm. All he has to do is to change the value of P_{th} when making a prediction with the model.

3.4 Results

The database of 30,441 losses has been randomly divided into a learning sample of 25,000 cases and a test sample with the remaining 10,441 cases. The decision tree boosting algorithm has been run on the learning sample and tested on the test sample. For a value of $P_{th} = 0.5$, the boosting model[2] misclassifies only 6.34% of the losses in the test sample[3]. This result is very good considering the large diversity of the topologies in the test sample and the fact that these topologies were not seen by the learning algorithm. For comparison, Veno and Westwood misclassify respectively 34.5% and 41.5% of the losses in the test sample. So, although these two simple models can still be good at predicting loss causes in some topologies, the boosting classifier is much better in average.

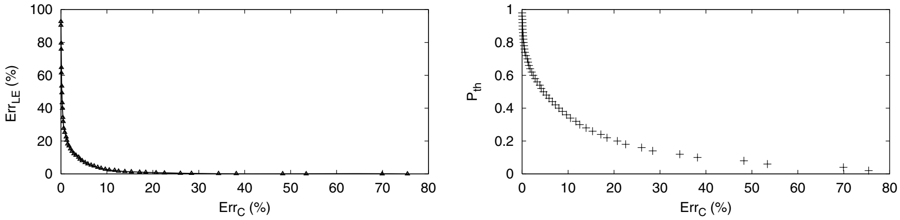


Fig. 1. At the left: The classification error obtained by Boosting. At the right: Err_C in function of P_{th}

We also evaluated the two types of errors Err_C and Err_{LE} on the test sample for different values of P_{th} going from 0.02 to 0.98 by step of 0.02. The left part

[2] The TCL code implementing this model is available at [4].

[3] For comparison, in [9], an artificial neural network yielded 7.7% error rate for this problem but with a much higher computational cost.

of Figure 1 plots Err_C in function of Err_{LE} when varying P_{th}. The closer the curve to the origin the better the model. We can see clearly that the curve is not far from the origin. There is clearly a tradeoff between the two types of error. It is possible to reduce the error of one class to zero but this is always at the expense of the other. One important thing to note is that we can decrease greatly the error on the classification of LE without increasing too much the error on the detection of congestions.

On Figure 1, the point corresponding to TCP, which has no mechanism to distinguish loss causes, is $(Err_C, Err_{LE}) = (0, 100\%)$. The operating points corresponding to the classification rules used by Veno [8] and by Westwood [15] are respectively $(54.29\%, 0.50\%)$ and $(63.20\%, 4.25\%)$. These results are again much worse than what we obtain with our approach. For example, for the same value of Err_{LE} as Veno and Westwood, boosting gives respectively an Err_C of about 22% (for $P_{th} = 0.18$) and 8% (for $P_{th} = 0.4$).

In terms of computing times for classification, it is clear that the boosting model is more expensive than Veno's or Westwood's rule. However, computing times remain very reasonable. Assuming the inputs have been computed, one classification with boosting requires about 250 simple comparisons. To give a rough idea, in our implementation[4], the classification of the 10,441 losses in the test sample requires about 340 msec, i.e. about 33μsec per classification. So, the computational cost of the classification should not be an issue in most cases.

4 Enhancement of Tcp with the Loss Classifier

Given a classification model for loss causes, we propose to modify TCP (Newreno) in the following way: When a loss is detected by triple duplicates, the result of its classification by the model is checked. If the cause is classified as congestion, the sender acts normally (i.e. it divides its congestion window by two), otherwise it maintains its congestion window constant.

However, we have at our disposal, not only one, but several classification models corresponding to different tradeoffs between the two types of error (by changing P_{th}). So, the question now is which value of P_{th} should be chosen with the double goal of improving TCP in wireless case and maintaining TCP-Friendliness. To ensure TCP-Friendliness, it is sufficient to maintain Err_C very close to zero. But, when Err_C is very low, Figure 1 shows that the corresponding Err_{LE} is too high to allow our model to really improve TCP in wireless case. So, the solution is to determine the lowest Err_{LE} that still preserves TCP-Friendliness. Since the two errors evolve in opposite direction to a change in P_{th}, this is equivalent to determining the highest Err_C that preserves TCP-Friendliness. By definition, a protocol is considered TCP-Friendly if the ratio between its rate and that of a competing TCP belongs to $[1/K, K]$ with $K \leq 1.78$ [6]. So, the value of Err_C (and hence of P_{th}) should be chosen as the largest value that still fulfills this condition.

[4] The classifier is implemented in C and is run on a Pentium 4 2.4 GHz.

To determine the target value of Err_C, let us suppose that we run one TCP NewReno, referred to simply as TCP in the sequel, and one TCP NewReno equipped with our Boosting classifier on the same network path[5] and that both flows lose a proportion p of their packets. According to Padhye et al. [14], the throughput of the TCP flow is equal to:

$$B_{tcp} = \frac{1}{RTT\sqrt{\frac{2p}{3}} + T_0 min(1, 3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

with the assumption that there is no delayed acknowledgment ($b = 1$).[6]

The TCP equipped with our Boosting classifier is a normal TCP except that it reacts only to a proportion $p(1 - Err_C)$ of packet losses instead of p. Its throughput over the network path is then equal to:

$$B_C = \frac{1}{RTT\sqrt{\frac{2pY}{3}} + T_0 min(1, 3\sqrt{\frac{3pY}{8}}pY(1+32p^2Y^2)},$$

where $Y = 1 - Err_C$. So, using these equations, it is possible to compute, for a given RTT and loss rate p, the largest value of Err_C such that $\frac{B_C}{B_{tcp}} < K$. However, since RTT and p are changing with time, instead of choosing a fixed value of P_{th}, we propose to dynamically adapt the value of P_{th} to the current values of RTT and p.

To this end, after each loss, we compute the loss rate p obtained over the whole session. Then, from this estimation and the RTT, we compute the highest value of Err_C such that $B_C/B_{tcp} < K$. Once the bound on Err_C is found, we retrieve the value of P_{th} that provides an error on congestion lower than this bound. The correspondence between P_{th} and Err_C is obtained on our test set and is illustrated in the right part of Figure 1.

Since P_{th} is adapted dynamically after each loss, from now on, we refer to TCP equipped with this classifier as “TCP+Boosting-adapt”.

In all our experiments below, we have used a value of K equal to 1.15 instead of the standard value of 1.78. There are two reasons for that. The first one is that we consider that 1.78 is too high; a smaller value will provide better friendliness. The second reason is that we prefer a more conservative choice of K in our case. Indeed, the value of Err_C for a given P_{th} in Figure 1 is only an estimate from the test sample of the true value. Hence, for a given situation, the choosen value of P_{th} can, in practice, lead to a higher Err_C than expected. By adopting a lower K, we minimize the impact of such situation.

5 Simulations with the Learned Models

In this section, we evaluate TCP+Boosting-adapt in wireless and wired networks. The experiments in wireless networks show the gain we can obtain in this kind of

[5] We focus here only on TCP-Friendliness in the wired case.

[6] Using $b = 1$ in the formula was recommended in RFC-3448.

networks where most of the losses are due to link errors. The experiments related to wired networks concern the TCP-Friendliness and the bandwidth usage.

For comparison, we test also TCP equipped with the classification rule used by Veno, referred to as "TCP+Veno-classifier". We have chosen the classifier of Veno because it provides lower Err_C and Err_{LE} than the classifier of Westwood. All the experiments have been done with `ns-2`.

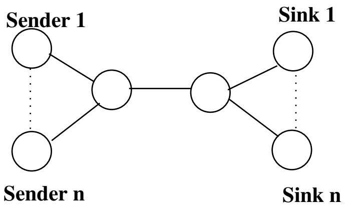


Fig. 2. Topology 1

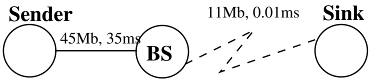


Fig. 3. Topology 2

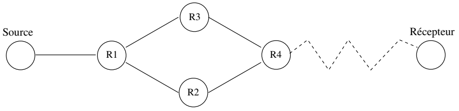


Fig. 4. Topology 3

5.1 The Improvement in Lossy Links

We test the hybrid topology used in [15] and illustrated in Figure 3. The first part, Sender-BS, is wired, and the second part, which connects the base station to the sink, is wireless. The bottleneck is the wireless link, which has a bandwidth equal to 11Mb/s. We compare the ratio between the throughput obtained by TCP+Boosting-adapt and the one obtained by TCP when we vary the packet loss rate from 0 to 5% over the wireless link. Each simulation is run 50 times. To have a good point of comparison, we run also simulations with an artificial TCP that classifies perfectly the cause of loss detected by triple duplicates. The graph at Figure 5 illustrates the ratio obtained by TCP with the three classifiers, the perfect one, the Veno classifier, and Boosting-adapt. Boosting-adapt is much better than the Veno classifier and also very close to the perfect model. Its gain with respect to TCP is not far from 300% when the loss rate is equal to 3%.

5.2 TCP-Friendliness and Link Capacity Usage

Wired Network. In this section, we compare the fairness of our protocol towards TCP in the wired case, which is an important criterion that should be fulfilled by the classification model. To test TCP-Friendliness, we use the topology illustrated in Figure 2 with $n = 2$, often used to test the fairness. The experiment consists of running TCP in competition with TCP+Boosting-adapt. Figure 6 illustrates the throughput obtained by each flow and shows that the share is fair.

For comparison, we have used TCP+Veno-classifier in the same scenario and the share ratio was slightly higher than five. This is not surprising since the Veno classifier is very bad at detecting congestion losses (its Err_C is high) and hence, it reduces its bandwidth less often than TCP.

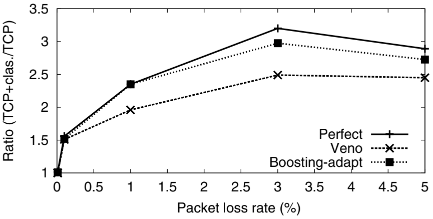


Fig. 5. The gain in lossy links

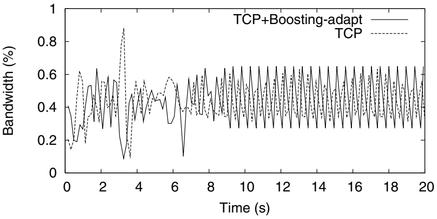


Fig. 6. TCP-Friendliness in wired case

Link Capacity Usage. Too much misclassifications of congestion losses can also lead to the underuse of the link when several TCPs equipped with a loss classifier compete over the same bottleneck. Indeed, if the modified TCPs do not react to packet losses due to a real congestion, then the congestion will actually worsen, leading to timeout expirations and thus to less throughput in average than with a standard TCP. To show that TCP+Boosting-adapt does not suffer from link underuse, we have used an aggregate of 4 similar flows, competing over one link (topology of Figure 2 with $n = 4$) and have computed their throughput and goodput. For comparison, we have also run an aggregation of TCP+Veno-classifier in the same situation. The results are given in Table 1. We can see that the throughput and the goodput of TCP+Boosting-adapt exceed those of TCP while those of TCP+Veno-classifier are much lower.

Table 1. The bandwidth usage of TCP with different classifiers

	TCP	TCP+Veno-classifier	TCP+Boosting-adapt
Throughput(%)	95.06	93.90	97.82
Goodput (%)	92.50	86.96	93.18

Route Change and Network Reordering. Route changes and failures have not been taken into account in the generation of the database, but we think that they will not affect the robustness of our approach. Indeed, our model has a memory of 2 RTTs (for the computation of the inputs). Thus, in the worst case, the classifier will misclassify all wireless losses happening during the two RTTs following the route change (just like a standard TCP). After this transition, the path will become “stable” again and the classifier will retrieve its ability of discriminating loss causes. For the same reason, network reordering was not taken into account during the generation of the database.

To confirm our statements, we use the topology of Figure 4 where our protocol is used between the sender and the sink and where the link R2-R4 breaks down after 15 seconds. We study the cases where the new path is longer, shorter and of the same length as the old one. The graphs in Figure 7 show the packets lost in the three cases with the misclassified losses represented by a square.

In the three cases, the discrimination quality is not deteriorated after the route change. In the case of a shorter new path, we have observed a case of reordering. Some packets arrive to the sink before their predecessors and they lead to triple duplicates at the sender side. The sender uses the classifier and concludes that the "loss" is not due to a congestion and then does not decrease its rate.

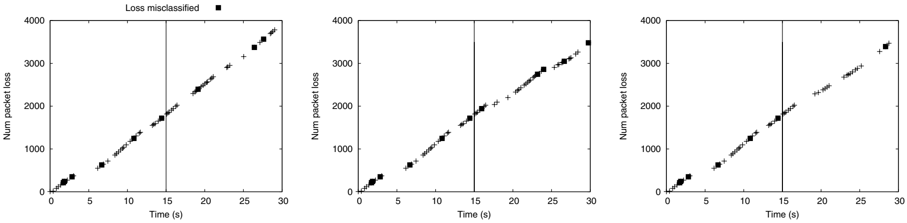


Fig. 7. Classification of losses when path changes (vertical line). From left to right: shorter, equal, and longer paths

6 Conclusion

In this paper, we have applied a supervised learning algorithm, called decision tree boosting, to automatically infer a loss cause classifier from a database of losses observed in a large number of random topologies. The resulting classifier has shown very good accuracy at classifying losses in random topologies that were not seen by the learning algorithm. Then, we have proposed to use this loss classifier to improve the performance of TCP in wired/wireless networks. To this end, we have shown how to adapt dynamically the classifier to ensure TCP friendliness. The new protocol, called TCP+Boosting-adapt, has shown a very good behaviour in wireless networks. It offered a high gain in throughput over wireless links and, at the same time, it preserved TCP-Friendliness in all topologies it has been tested over.

We see two potential limitations to our approach. First, we have not taken into account losses detected by timeout expiration, which were thus all considered as signs of congestion. Our classifier is thus not relevant for short TCP sessions where timeout expiration is the only mechanism to detect congestion. However, even without classifying such losses, the throughput gains observed in our simulations are already excellent and we can only improve these results by taking into account time out expirations when designing the loss classifier. Furthermore, congestions are not so harmful for short sessions which lose little throughput in the case of packet losses.

A second potential limitation is that our database was generated from simulated networks (topologies and traffics) which may differ to a certain extent from actual ones. However, the learning sample was generated by randomizing all networks conditions. Hence, there is no bias in the classifier. We have also carried out experiments with a more realistic topology generator (BRITE [13]) that have not shown significant differences in terms of performance of the classifiers with respect to the results presented in this paper. Furthermore, restraining the learning to actual topologies and flows can only improve the accuracy of the classification.

Acknowledgements

This work has been partially supported by the Belgian Science Policy in the framework of the IAP programme (MOTION P5/11 project) and by the E-NEXT European Network of Excellence (NoE). P.G. is a Postdoctoral Researcher at the National Fund for Scientific Research (FNRS, Belgium).

References

1. A. Bakre and B. R. Badrinath. I-tcp: indirect tcp for mobile hosts. In *Proc. of the 15th Int. Conf. on Distributed Computing Systems*, 1995.
2. S. Biaz and N. H. Vaidya. Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result. In *Proc. of IC3N, New Orleans*, 1998.
3. L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth International (California), 1984.
4. I. El Khayat, P. Geurts, and G. Leduc. Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning. *Submitted*, 2005. http://www.run.montefiore.ulg.ac.be/ elkhayat/Boosting-DT/Boosting-dt.html.
5. S. Floyd. A report on some recent developments in TCP congestion control. *IEEE Communication Magazine*, 39(84-90), April 2001.
6. S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of SIGCOMM 2000*, pages 43–56, 2000.
7. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of the 2th European Conference on Computational Learning Theory*, pages 23–27, 1995.
8. C. P. Fu and S. C. Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE JSAC*, February 2003.
9. P. Geurts, I. El Khayat, and G. Leduc. A machine learning approach to improve congestion control over wireless computer networks. In *Proc. of IEEE Int. Conf. on Data Mining (ICDM-2004)*, pages 383–386, 2004.
10. A. Gurtov and S. Floyd. Modeling wireless links for transport protocols. *SIGCOMM Computer Communication Review*, 34(2):85–96, 2004.
11. T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction.* Springer, 2001.
12. J. Liu, I. Matta, and M. Crovella. End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
13. A. Medina, I. Matta, and J. Byers. BRITE: A Flexible Generator of Internet Topologies. Technical report, Boston University, 2000.
14. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, 2000.
15. R. Wang, M. Valla, M.Y. Sanadidi, B.K.F Ng, and M. Gerla. Efficiency/Friendliness Tradeoffs in TCP Westwood. In *Proc. of the 7th IEEE Symposium on Computers and Communications*, 2002.

A Multiple Time-Scale Model for TCP Bandwidth Sharing Under User Heterogeneity

Dirk Abendroth[1,2], Hans van den Berg[2,3], and Michel Mandjes[2,4]

[1] Technical University Hamburg-Harburg,
Denickestr. 17, 21073 Hamburg, Germany
[2] University of Twente, P.O. Box 217,
7500 AE Enschede, The Netherlands
[3] TNO Telecom, Brasserplein 2, P.O. Box 5050,
2600 GB Delft, The Netherlands
[4] CWI, Kruislaan 413, P.O. Box 94079,
1090 GB Amsterdam, The Netherlands

Abstract. Building on the vast body of existing TCP models, we develop a novel versatile model that explicitly captures user heterogeneity, and takes into consideration dynamics at both the packet level and the flow level. It is described how the resulting multiple time-scale model can be numerically evaluated. Validation is done by using NS2 simulations as a benchmark. In extensive numerical experiments, we study the impact of heterogeneity in the round-trip times on user-level characteristics such as throughputs and flow transmission times, thus quantifying the resulting bias. In particular, we investigate to what extent this bias is affected by the networks' 'packet-level parameters', such as buffer sizes. We conclude by extending the single-link model in a straightforward way to the general network setting.

Keywords: TCP, user heterogeneity, performance, throughput, round-trip times, packet level, flow level.

1 Introduction

Enabled by enhanced access technologies such as 'fiber to the home', ADSL, wireless LANs or UMTS, the number of users with high-speed access to the Internet is increasing rapidly. At the same time, more and more Internet applications require some sort of minimum quality-of-service (QoS), expressed in terms of delay, throughput, etc. With most of the data transfers relying on TCP as underlying end-to-end transport protocol, the performance of TCP-controlled networks has become a prominent theme in networking research.

TCP has been designed to support efficient and reliable transmission of elastic data flows, tolerating some variations in the throughput. In particular, based on implicit information about the level of network congestion (round-trip time, packet loss) TCP increases or decreases the sending rate in order (to attempt) to provide a fair share of the network resources to all users. However, it is clear that heterogeneous user behavior might lead to asymmetries in the experienced performance. For instance, one may wonder to what extent it pays off to have a higher access rate or a shorter round-trip time

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 561–573, 2005.

than the other TCP flows. It is this relation between user heterogeneity and flow-level performance (throughput, flow transfer times) that is studied in this paper.

A study on the impact of user heterogeneity could be done relying exclusively on simulation tools (like the NS2 [13] TCP simulator), but such an approach has its well known inherent limitations; in particular, (basic) simulation methods hardly allow for doing sensitivity analysis, as it may already be rather time-consuming to get a reliable estimate for a single parameter instance. Therefore we have chosen to set up a model that allows for an analytical approach, or, when its evaluation turns out to be too complicated, a hybrid approach in which the role played by simulation is minimized.

As motivated by the above, TCP's widespread use and complex behavior have triggered the search for simple and transparent, yet accurate, mathematical techniques for performance analysis. Many performance models have been proposed, which can be roughly divided into *packet-level models* and *flow-level models*.

Packet-level models describe the detailed dynamics of TCP, related to the evolution of the flows' window sizes and transmission rates. Evidently, the detailed behavior of a link fed by various classes of TCP flows, is intrinsically difficult to capture, particularly due to the complex interactions between the source behavior and the congestion level. However, by assuming a constant number of greedy (i.e., persistent) flows, and after imposing some additional simplifications, explicit expressions were obtained for the flows' throughput, as a function of the packet loss probability; early references are Kelly [15], Mathis *et al.* [21], and Padhye *et al.* [23]. Noticing that in return the packet loss probability of a bottleneck link is a function of the offered load (and hence essentially also of the throughput), we obtain two equations in two unknowns, and as a result the throughput can be found, see e.g. [3, 8, 17]. Some papers explicitly model TCP's window dynamics, and allow at the same time user heterogeneity, e.g., [11, 22], but this approach is severely limited by scalability constraints.

Where packet-level models consider a constant set of persistent flows feeding into a link, flow-level models explicitly focus on the dynamics of the number of flows present. In other words: flow-level models focus on a somewhat less detailed time-scale than packet-level models, namely the time-scale at which flows arrive and depart, see for instance [20]. It is assumed that, at the moment such a flow-level transition takes place, the allocation of the transmission rates to the individual flows adapts instantly. This enables the use of *processor sharing* (PS) queues, addressed in great generality by Cohen [9]. Over the past years this class of models has gained ground as a generic description of TCP's flow-level dynamics, as advocated in, e.g., [5, 16].

It is clear that TCP's packet-level and flow-level have a strong mutual dependency, which motivates the attempts to develop a unified approach. In this respect we mention the pioneering work of Gibbens *et al.* [12], who consider the network extension of the single-link packet-level models mentioned above, enabling them to compute the throughputs for any given number of flows simultaneously present at the various routes through the network. Then they weigh these throughputs with an *a priori* supposed distribution for this number of 'concurrent flows' (Poisson, geometric), in order to 'emulate' the flow level. Though reasonable, a rigorous justification of assuming these specific flow-level distributions was lacking. This motivated why Lassila *et al.* [19] have considered ways to *derive* (i.e., to 'endogeneously determine') the flow-level distribution from the model, rather than to exogeneously impose a distribution. It is noted that [19] has succeeded in doing so in a single-link setting with homogeneous input (i.e., flows have the same job-size distribution, round-trip times, and access rates).

In the present paper we build on the results of [12, 19], but we add a number of substantial enhancements. Relative to [19], a first contribution of our work is that we allow for heterogeneous input at the flow level: several classes are distinguished (characterized by flow arrival rate, flow size, round-trip time, and access rate). Then the procedure is that we first apply packet-level models to compute the (per-class) throughputs for a fixed number of flows present, which we use as the input for a PS-type of flow-level model. It is noted that the PS model that arises in this setting has the flavor of a so-called *discriminatory processor sharing* (DPS) system, which is in general notoriously hard to study analytically, see, e.g., [10]. A second improvement over [19] is that our multiple time-scale framework lends itself to being extended to network settings. The model can be used to assess the effect of user heterogeneity in often highly complex multi-link situations. Compared to [12] the major improvement is that we, as in [19], *derive* the flow-level distribution (i.e., the joint distribution of the number of 'concurrent flows' of the various types), rather than that we impose *a priori* some distribution.

It is noted that, particularly in the situation that the number of user classes grows large, it may become extremely time-consuming to numerically solve the flow level. For those situations we propose a 'hybrid' method: the packet-level is solved numerically, and the resulting throughputs are used as input in the *simulation* of the flow level. It is noted that such an approach is still substantially faster than detailed time-scale (NS2) simulations, as only the flow-level jumps need to be simulated rather than the full packet-level dynamics. We remark that a somewhat similar hybrid approach was developed in [4]; there the simulated jumps correspond to congestion epochs.

The remainder of this paper is organized as follows. In Section 2 we introduce our packet/flow level model for heterogeneous TCP-controlled traffic transmitted over a buf-fered link. Quantitative validation of the model, as reported in Section 3, is done by using the simulator NS2 [13] as a benchmark. Also a number of other numerical experiments are performed, with a strong focus on the impact of the heterogeneity in round-trip times on the throughputs experienced by the various user classes. We give some further comments on the relation with discriminatory processor sharing. Section 4 describes the extension of the single-link model of Section 2 to non-cyclic multiple-link network scenarios. Importantly, it is shown that 'packet-level parameters' such as buffer sizes, do have a significant impact on the way TCP allocates bandwidth. Section 5 concludes the paper.

2 Integrated Packet/Flow Level Modelling Approach

In this section we introduce a mathematical model that describes a buffered network link fed by a fluctuating set of heterogeneous TCP flows. As argued in the introduction, a model that captures all the details is far too complex to analyze.

A commonly used resort is to rely on *time-scale decomposition*: distinguish between multiple time-scales, solve these separately, and integrate them into the performance measures of interest. We here follow such a decomposition approach, by decoupling the *packet level* and the *flow level*. Recall that the packet level describes the performance when the link is used by a fixed set of persistent flows, whereas the flow level describes the fluctuations of the number of flows simultaneously present. Interestingly, the decomposition allows us to analyze the effect of typical packet-level parameters (buffer size, round-trip times, etc.) on flow-level performance (the average number of flows in the system, flow transfer times, etc.). It is stressed that our specific focus is on assessing the impact of user heterogeneities on the performance. To this end, we will introduce m classes of flows.

Modelling Assumptions. Each class i, for $i = 1, \ldots, m$, is characterized by four parameters. (i) The rate at which flows of class i are initiated is denoted by λ_i; it is assumed that these arrivals follow a Poisson process. (ii) Flows of class i have an exponentially distributed size with mean μ_i^{-1} (packets); it is assumed throughout this paper that packets are equally sized. (iii) The 'physical' round-trip time of a class-i packet, i.e., due to propagation and all other non-congestion-dependent factors, is given by RTT_i^0 time units (in other words: the queueing delay in the buffer is *not* incorporated in RTT_i^0). (iv) The access rate of a class-i user is given by R_i (packets per time unit). Notice that the maximum window size $W_{\max,i}$ and the round-trip time also put a limit on the users' transmission rate: $R_i \leq W_{\max,i}/\mathrm{RTT}_i^0$.

The link, to be interpreted as a *bottleneck* link, is characterized by its service speed (or link rate) C (expressed in packets per time unit) and buffer size B (packets). The service discipline is *first-in-first-out*. The m traffic classes share these common network resources. For reasons of stability, the incoming load is limited to the system's capacity: $\sum_{i=1}^m \lambda_i/\mu_i < C$. Now we subsequently describe the packet level and flow level, and describe how these allow the computation of the performance measures of our interest.

Packet Level. In the packet-level model there is a *fixed* number n_i of *persistent* TCP flows of class i, for $i = 1, \ldots, m$. The main objective of the packet-level analysis is to compute, for a given vector $\bar{n} := (n_1, \ldots, n_m)$, the throughputs $t_1, \ldots, t_m$ of the various classes.

In our approach we rely on a relation between the mean throughput, the packet-loss probability, and the round-trip time, that was derived for the case of just one class of TCP connections, see e.g. [15, 21, 23]; in our study we will rely on the formula given in [15]. Suppose there are n flows with round-trip time RTT and access rate R, who experience a packet-loss probability p. Then the (total) throughput in the congestion-avoidance phase is approximated by

$$t(n) = \min\left\{ nR, \frac{n}{\mathrm{RTT}}\sqrt{\frac{2(1-p)}{p}} \right\};$$

evidently, the left-hand argument of the $\min$ function simply limits throughputs to the access rate R. By borrowing the above formula, we obtain for our case of m heterogeneous traffic classes:

$$t_i(\bar{n}) = \min\left\{ n_i R_i, \frac{n_i}{\mathrm{RTT_i}}\sqrt{\frac{2(1-p_i)}{p_i}} \right\}, \qquad i = 1, \ldots, m. \tag{1}$$

It is clear that the round-trip time RTT_i consists of its fixed component RTT_i^0, increased by the queueing delay experienced by class i. With δ_i denoting the mean queueing delay of class i, we could write $\mathrm{RTT}_i = \mathrm{RTT}_i^0 + \delta_i$. Regarding loss and delay, it seems reasonable to assume that there will not be too much discrepancy between the classes, which allows us to write $\delta_i = \delta$ and $p_i = p$, for $i = 1, \ldots, m$.

Having expressed the throughput in terms of the packet loss and delay, we have to find loss and delay as a function of the throughput; clearly, having these relations at our disposal, we are able to compute the $t_i(\bar{n})$. To this end, we make the approximation that, at the packet level, packets (of equal size) arrive at the buffer according to a Poisson process. Hence, we can approximate the packet-loss probability and the mean delay by those of the corresponding M/D/1/B queue. In self-evident notation:

$$p \equiv p(\bar{n}) = p_{\mathrm{M/D/1}}\left(\sum_{i=1}^{m} t_i(\bar{n}), C, B\right); \quad \delta \equiv \delta(\bar{n}) = \delta_{\mathrm{M/D/1}}\left(\sum_{i=1}^{m} t_i(\bar{n}), C, B\right). \qquad (2)$$

Techniques for computing (or approximating) both the loss probability and mean delay in M/D/1 queues are described in, e.g., [24–Section 15.1]; the mean delay is given by their Eq. (15.1.2), whereas the loss probability could be accurately approximated by exponential expansion in the spirit of their Eq. (15.1.6).

The assumption of Poisson arrivals at the packet level has been very common in the literature, see e.g. [3, 12, 17]. Given the fact that many flows are multiplexed, and in the absence of a detailed description of the packet arrival process, we have chosen to do so. It can obviously not be justified that this choice yields the best match with the actual queueing behavior. We emphasize, however, that the methodology presented here does not critically rely on this Poisson assumption; if there is a reason to assume that some other arrival process provides a better match, one could employ the corresponding queueing model instead.

It is clear that inserting (2) into (1) yields a fixed-point problem of m non-linear equations with m unknowns (i.e., $t_1, \ldots, t_m$; we suppress the $\bar{n}$ for convenience). Now consider the right-hand side of (1) as a function of t_i, for fixed t_j, $j \neq i$. Evidently, both the mean queueing delay δ and the loss probability p increase in t_i. It can be verified easily that this implies that the right-hand side of (1) decreases in t_i, and has limit 0. As the left-hand side (i.e., the identity function) increases from 0 to ∞, we conclude that there is, for given t_j, $j \neq i$, a fixed point.

Flow Level. The flow level describes the dynamics related to arrivals and departures of flows (i.e, TCP connections). When there are $\bar{n}$ flows in the system, we assume that class i is served at a rate $s_i(\bar{n}) := t_i(\bar{n})(1 - p(\bar{n}))$, where $t_i(\bar{n})$ and $p(\bar{n})$ follow from the packet-level fixed-point equations; $s_i(\bar{n})$ is often referred to as the class-i 'goodput'. All individual flows of type i are assumed to get a 'fair share' $s_i(\bar{n})/n_i$ of the class-i goodput. It is remarked that our time-decomposition entails that we implicitly assume that, when the number of active flows changes, service rate adaptation takes place instantly. Put differently, our approach neglects the rate fluctuations at the packet level (which are due to the window-size dynamics, in response to packet losses).

As the flow sizes were assumed to be exponential, we can model the flow-level dynamics by a (continuous-time) Markov chain. The fact that any flow gets a fair share of the per-class goodput, gives our model the flavor of a processor-sharing system. The single-class system can be solved explicitly; see [19], relying on the results of [9]. The user heterogeneity, as present in our model, however, makes the analysis considerably more difficult. As mentioned in the introduction, the flow-level model that arises in this setting has the flavor of a DPS system, a class of models that is in general hard to study analytically.

We now give the transition rates of the continuous-time Markov chain governing the flow level, with state space $\mathbb{N}_0^m$. The transition rate corresponding to an arrival of a type-i flow, denoted by $q(\bar{n} + e_i \mid \bar{n})$, is clearly equal to λ_i; here $e_i := (0, \ldots, 0, 1, 0, \ldots, 0)$, where the 1 is put on the ith position. It can be verified easily that the transition rate from $\bar{n}$ to $\bar{n} - e_i$, i.e., $q(\bar{n} - e_i \mid \bar{n})$, equals $\mu_i s_i(\bar{n})$ (under the proviso that $n_i > 0$; otherwise $\bar{n} - e_i$ clearly does not belong to the state space). Having these transition rates, we can compute the equilibrium distribution $\pi(\bar{n})$.

In case solving the system of balance equations leads to numerical problems (the iterative solution techniques may yield oscillations), these can be overcome by using simulation at the flow level. This procedure could be characterized as a *hybrid* approach, between computation and simulation. An alternative could be to simulate *both* packet-level and flow-level dynamics, for instance by using NS2, but such an approach is substantially slower. Clearly, the main difference lies in the fact that our integrated approach does not need to simulate the detailed packet-level dynamics; these are summarized by the (computed) throughput values. In this respect there is an interesting similarity with the hybrid simulation/computation approach proposed in [4]; an important difference is that in our approach the flow-level fluctuations (flow arrivals, flow departures) are the jump epochs, whereas in [4] this role is played by the congestion epochs.

3 Validation; Impact of User Heterogeneity

This section presents the quantitative validation of our packet/flow level approach by using NS2 simulations; we have chosen to use NS2 [13] as a benchmark, motivated by the fact that it mimicks TCP at a detailed time-scale. Hence, comparing results obtained by using our methodology with results obtained under NS2, we assess to what extent it is justified to replace the packet-level dynamics by a static bandwidth allocation according to the rates $s_i(\bar{n})$ (when there are $\bar{n}$ flows in the system), as is done in our approach.

We also perform a number of further experiments assessing the impact of heterogeneity on the performance bias (i.e., the difference between the classes in, e.g., the mean file transfer delay) with emphasis on the impact of packet-level parameters (such as the buffer size).

Scenarios. We consider the single bottleneck scenario as in Section 2, with $m = 2$ traffic classes. Recall that the number of concurrently active flows within each class (n_1 and n_2) is varying over time due to flow arrivals and departures.

In our experiments we chose the loads per class 1 and 2 to be equal: $\lambda_1/\mu_1 = \lambda_2/\mu_2 = \rho/2$, where $\rho < C$ is the system load. We concentrate on the heterogeneity with respect to round-trip times and file sizes; we assume that all connections have the same access link rate, i.e., $R_1 = R_2$. We set the link rate C equal to 10 [Mbit/s]. All transmitted packets have a size of 1500 [bytes] (cf. MTU size).

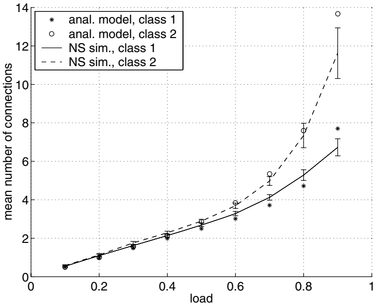


Fig. 1. Mean number of connections for two classes, under varying load. Access rates 1 [Mbit/sec.], $\mathrm{RTT}_1^0 = 100$ and $\mathrm{RTT}_2^0 = 200$ [msec.], buffer size 10 [packets]

In practical situations round-trip times vary, roughly speaking, over a range of 50 to 200 msec. Therefore we have chosen to consider round-trip times of 50, 100, and 200 msec. We study the influence of small and large buffer sizes, i.e., 10 and 50 [packets], and access rates, i.e., 1 and 2 Mbit/sec. In the sequel we provide a comparison of results for a full permutation of all the above parameters, where, in addition, both classes have different round-trip times. We have set the mean file size to 500 [packets] for all flows. Recall that we took a constant packet size of 1500 bytes.

Numerical Results. For different values of the system load ρ, Figure 1 shows the mean number of active connections per class $n_i^\star$; the graph contains both the values obtained by using our method, and the values obtained by using NS2 (including their confidence intervals).

In Figures 2 and 3 we have focused on the perhaps somewhat more appealing performance measure of the mean flow transfer delay D_i, related to $n_i^\star$ through Little's law $D_i = n_i^\star/\lambda_i$. Bearing in mind the complex TCP dynamics, the figures show that our model results coincide remarkably well with the NS simulation results, and it does so for a wide set of parameter combinations.

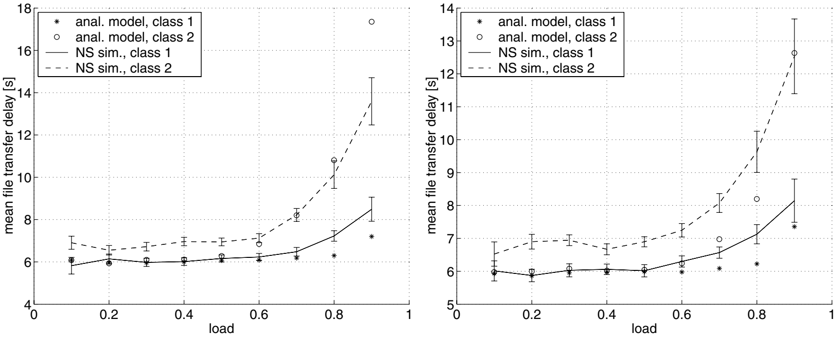


Fig. 2. Mean file transfer delay [sec.]; access rates 1 [Mbit/sec.], $\mathrm{RTT}_1^0 = 50$ and $\mathrm{RTT}_2^0 = 200$ [msec.], buffer size left panel: 10 [packets], buffer size right panel: 50 [packets]

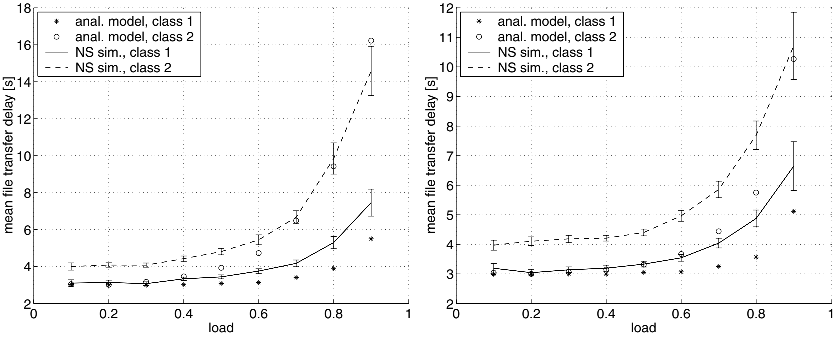


Fig. 3. Mean file transfer delay [sec.]; access rates 2 [Mbit/sec.], $\mathrm{RTT}_1^0 = 50$ and $\mathrm{RTT}_2^0 = 200$ [msec.], buffer size left panel: 10 [packets], buffer size right panel: 50 [packets]

We now discuss in detail the accuracy of our packet/flow level modelling approach. We mention the following observations:

- For low values of the load, our model predicts nearly identical performance for class 1 and class 2. The NS2 simulations, however, show that these results are systematically too optimistic and that the flows with the short round-trip time experience the smaller delay. We suspect that this behavior is caused by the fact that our method does not incorporate the effects of TCP's slow-start phase: the time-scale decomposition entails that we do as if the flow is, after being initiated, immediately in its 'stationary behavior'. It is noted that, during the slow-start phase, the round-trip times play a crucial role, as they dictate how fast the window size can grow. In [19] the effects of the slow-start phase were successfully compensated, in order to avoid too optimistic estimates; one could pursue the development of such a procedure for the model under study.
- When comparing Fig. 2 (access rate of 1 Mbit/sec.) with Fig. 3 (access rate of 2 Mbit/sec.), it could be concluded that for low values of the load the transfer delays are not so much determined by the round-trip times, but rather by the access rate constraints. This is reflected by the fact that the transfer delays in Fig. 2 are (roughly) two times as high as in Fig. 3.
- The numerical results (also the ones not shown here) indicate that our model (particularly for class 2) is less accurate for large buffers ($B = 50$) than for small buffers ($B = 10$). It is expected that this is due to the fact that the real packet arrival process is often substantially burstier than Poisson, leading to errors in the approximation of the loss probability p and queueing delay δ, which are typically smaller for a small buffer than for a large buffer.

Refinements of our approach, which take into account the issues identified above, are subjects for future research.

Quantification of the Bias; Equalizing Effects. We finally say some words on the quantification of the bias, in the situation of heterogeneous round-trip times. There are

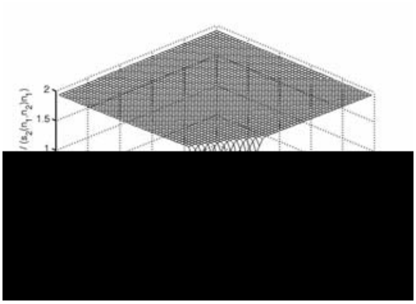


Fig. 4. 'Normalized goodput ratio' $(s_1(n_1,n_2)/n_1)/(s_2(n_1,n_2)/n_2)$ plotted as function of the states (n_1,n_2); access rates 1 [Mbit/sec.], $\mathrm{RTT}_1^0 = 100$ and $\mathrm{RTT}_2^0 = 200$ [msec.], buffer size 10 [packets]

Table 1. 'Normalized goodput ratio', buffer sizes are in packets

	$B = 10$	$B = 50$
$r = 2$	1.90	1.63
$r = 4$	3.45	2.38

several papers on this issue, see for instance [1, 18]. We here study the claim that per-flow throughputs are inversely proportional to the flow's round-trip time, in line with formula (1), see, e.g., [15, 21, 23]. In case of two classes sharing a bottleneck link, this would mean that

$$\frac{s_1(\bar{n})/n_1}{s_2(\bar{n})/n_2} \approx \frac{\mathrm{RTT}_2}{\mathrm{RTT}_1}. \tag{3}$$

Consider a situation with $\mathrm{RTT}_2^0/\mathrm{RTT}_1^0 = 2$. We plot the left hand side of (3), by using the numbers obtained in our packet-level model, see Figure 4.

Indeed, the ratio is nearly constant for somewhat larger values of n_1, n_2, and has indeed value 2; for 'low' states the ratio is significantly smaller. In fact, close to the origin the ratio is nearly 1, as such a small number of connections (with limited access rate) is not able to 'fill up the link'.

There are circumstances under which the performance asymmetries tend to be 'equalized'. In the first place, this is clearly the case for low and medium loads, and limited access rates. For such loads the most probable states are relatively close to the origin, and Figure 4 then indicates that the 'normalized goodput ratio' will be close to 1.

A second factor that has impact on this 'equalizing effect' is the buffer size. Table 1 shows the limiting value of the 'normalized goodput ratio', i.e., $(s_1(\bar{n})/n_1)/(s_2(\bar{n})/n_2)$ for large n_1, n_2. We do so for varying (i) ratio of the ('physical') round-trip times $r := \mathrm{RTT}_2^0/\mathrm{RTT}_1^0$, and (ii) buffer size. The link capacity is still 10 [Mbit/sec.].

From the table we conclude that for small buffers the 'normalized goodput ratio' is close to the ratio of the round-trip times, whereas for larger buffers this match is less good. An explanation lies in the fact that for larger buffers the round-trip times are increasingly determined by the queueing delay (rather than the 'physical' delay), and this queueing delay is the same for both classes.

We conclude that it is not always accurate to assume goodputs inversely proportional to the round-trip time. It also entails that it could be inaccurate to approximate the model with heterogeneous users with discriminatory processor sharing [10], with weights inversely proportional to the classes' round-trip times, as done in, e.g., [2]. Our model nicely captures the 'equalizing effect' of the access rate limitation and the buffer size on the throughputs, as experienced by flows with different ('physical') round-trip times; it is noted that similar properties were observed in the simulations performed in [5].

4 Extension to Networks

In this section we make a first step towards extending our single-link framework to a network setting. Model and analysis are presented in Section 4.1. The approach borrows elements of [12], but we remark again that the crucial difference between our approach and [12] is, that our model finds the distribution of the network population 'endogeneously'. In Section 4.2, we consider a few examples, and investigate how these compare to other results on rate allocation.

Modelling and Analysis. Consider a non-cyclic network, consisting of k links, characterized by their link rates C_j and buffer sizes B_j. Flows of class i subsequently pass a number of links. We say that $j \in \mathcal{R}_i$ if class i uses link j. Also, we say that $j_1 \prec_i j_2$ if $j_1, j_2 \in \mathcal{R}_i$, and in addition j_1 lies on this route *before* j_2.

The approach relies again on decoupling the packet level and the flow level. We first determine the throughput of class i, for a given user population $\bar{n}$; then this serves as input for the continuous-time Markov chain model that describes the flow level. The generalization of the throughput formula (1) is, for $i = 1, \ldots, m$,

$$t_i(\bar{n}) = \min \left\{ n_i R_i \, , \, \frac{n_i}{\mathrm{RTT}_i^0 + \sum_{j \in \mathcal{R}_i} \delta_j} \sqrt{\frac{2\left(1 - \sum_{j \in \mathcal{R}_i} p_j\right)}{\sum_{j \in \mathcal{R}_i} p_j}} \right\} . \tag{4}$$

Here p_j is the loss probability at link j, and δ_j the mean delay at link j; we assume that the per link loss probability and mean delay is constant across the user classes.

In return, the loss probabilities and mean delays depend on the load offered to the links. We can write

$$p_j \equiv p_j(\bar{n}) = p_{\mathrm{M/D/1}} \left(\sum_{i: j \in \mathcal{R}_i} t_i(\bar{n}) \cdot \left(1 - \sum_{k \prec_i j} p_k(\bar{n}) \right), C_j, B_j \right) ; \tag{5}$$

$$\delta_j \equiv \delta_j(\bar{n}) = \delta_{\mathrm{M/D/1}} \left(\sum_{i: j \in \mathcal{R}_i} t_i(\bar{n}) \cdot \left(1 - \sum_{k \prec_i j} p_k(\bar{n}) \right), C_j, B_j \right) . \tag{6}$$

Notice that the load imposed on link j is 'thinned', along the various routes i, by the losses in its predecessing links (i.e., k such that $k \prec_i j$); the formula uses, as in [12], the approximation that losses at the various links are independent, cf. also [14], and the well known approximation $\prod_i (1 - x_i) \approx 1 - \sum_i x_i$ for small x_i. Combining (5) and (6) with the throughput formula (4) again yields m equations in m unknowns, which can be solved numerically.

The flow level can again be done by constructing a continuous-time Markov chain; it is obvious that again it can be solved by either solving the balance equations, or by simulating an m-dimensional random process. It is noted that the flow-level procedure remains essentially the same, and that the network topology is just reflected by the packet-level computations.

Examples; Rate Allocation in TCP. It has been claimed that the transmission rates allocated in TCP's congestion avoidance are well approximated by the optimizing t_i, $i = 1, \ldots, m$ in

$$\min_{t_1,\ldots,t_m} \sum_{i=1}^{m} \left(\frac{1}{\mathrm{RTT}_0^i}\right)^2 \frac{n_i}{t_i} \qquad \text{under} \qquad \sum_{i:j\in\mathcal{R}_i} t_i \le C_j;$$

see for instance [6–Eq. (3)]. For a single link fed by m heterogeneous classes, when performing the optimization, one obtains, with κ_i defined as $(\mathrm{RTT}_i^0)^{-1}$, that $t_i(\bar{n})$ equals $Cn_i\kappa_i / \sum_{j=1}^{m} n_j\kappa_j$, i.e., the rate allocation is according to DPS with weights inversely proportional to the round-trip times. We have seen in the previous section that this approximation is particularly accurate when buffers are relatively small.

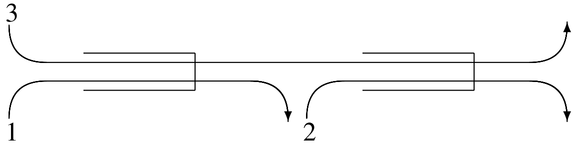


Fig. 5. Topology of two-link example

Now consider the somewhat more involved case of a network with multiple links. For ease we concentrate on the simplest, non-trivial model, as depicted in Figure 5, since this already shows a number of features that are not present in the single-link case. Let there be two links, both of rate C; let type 1 use link 1, type 2 use link 2, and type 3 use both links. It can be verified that the above minimization now yields that $t_1(\bar{n}) = t_2(\bar{n})$ and

$$t_1(\bar{n}) = \frac{C \cdot \sqrt{(n_1\kappa_1)^2 + (n_2\kappa_2)^2}}{n_3\kappa_3 + \sqrt{(n_1\kappa_1)^2 + (n_2\kappa_2)^2}}; \quad t_3(\bar{n}) = \frac{C \cdot n_3\kappa_3}{n_3\kappa_3 + \sqrt{(n_1\kappa_1)^2 + (n_2\kappa_2)^2}}. \qquad (7)$$

We see that this allocation rule is not quite DPS, and we wonder whether it coincides with the throughputs realized in our model and in NS2. Interestingly, even when $\mathrm{RTT}_1^0 \neq \mathrm{RTT}_2^0$, the above allocation indicates that the flows of type 1 enjoy the same throughput as the flows of type 2. In practice, however, one expects that this heterogeneity of round-trip times should have impact.

To study these effects we have performed two experiments. In the first we vary the heterogeneity between the round-trip times. This is done by choosing $\mathrm{RTT}_1^0 = 1$ and $\mathrm{RTT}_3^0 = 3$, and $\mathrm{RTT}_2^0 \in \{0.5, 1, 2, 5, 10\}$. We have fixed the numbers of users $n_1 = n_2 = n_3 = 10$, the buffer size $B_1 = B_2 = 10$ packets, and the link rates $C_1 = C_2 = 100$. For ease we assume that no access rate limitations are imposed. The

Table 2. A. (left) Throughputs $t_i(\bar{n})$ as a function of RTT$_2^0$; the allocation according to (7) is given between parentheses. B. (right) Throughputs $t_i(\bar{n})$ as a function of B; the allocation according to (7) would be $t_1 = 75.4$, $t_2 = 75.4$, and $t_3 = 24.6$.

RTT$_2^0$	t_1	t_2	t_3
0.5	72.4 (87.0)	77.1 (87.0)	12.2 (13.0)
1	68.3 (80.9)	68.3 (80.9)	16.7 (19.1)
2	65.6 (77.0)	60.7 (77.0)	19.8 (23.0)
5	64.3 (75.4)	53.0 (75.4)	21.3 (24.6)
10	64.0 (75.1)	48.1 (75.1)	21.6 (24.9)

B	t_1	t_2	t_3
2	43.9	22.3	13.7
5	56.9	39.9	18.5
10	64.2	52.9	21.3
20	68.4	62.0	24.6
50	69.9	67.2	26.5

throughputs as derived by our model, as well as those based on (7), are tabulated in Table 2.A. We conclude that the heterogeneity does have a significant impact on throughputs; in particular, there could be a substantial difference between t_1 and t_2, which was not predicted by the rate allocation models of, e.g., [6]. Interestingly, class 3 considerably benefits when RTT_2^0 increases.

The second experiment studies the impact of the buffer size on the bias. We have varied $B = B_1 = B_2$, and fixed $\mathrm{RTT}_0^2 = 5$; the other parameters are the same as in the first experiment. Table 2.B shows the same 'equalizing effect' as we have seen before: for small buffers the bias is very pronounced, whereas for larger buffers, class 1 and class 2 see essentially the same round-trip delay (mainly queueing delay), and as a consequence the bias disappears.

5 Concluding Remarks

We have developed a versatile TCP performance model that explicitly captures user heterogeneity. This multiple time-scale model integrates the dynamics at the packet level with those at the flow level. It is relatively simple, and allows for straightforward numerical evaluation. The accuracy of the model was validated by using TCP simulator NS2. In extensive numerical experiments, we have studied the impact of heterogeneity in the round-trip times on user-level characteristics (throughputs, flow transmission times). Interestingly, we have seen that the asymmetry caused by this heterogeneity is somewhat mitigated if the access rates are small (and, in addition, the load is relatively low), or the buffer is large.

We have pointed out how the single-link model can be extended in a straightforward way to the general network setting. It is noted that settings with heterogeneous users (in particular networks) pose interesting questions related to rate allocation. Like in the single-link case, we have seen that the size of the buffers does affect the rates allocated by TCP: again for large buffers the performance bias disappears. The network model opens up the possibility of a careful assessment of those phenomena; it is remarked that most existing rate allocation results, see, e.g. [6], do not take into account buffering.

References

1. E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange. Fairness analysis of TCP/IP, Proc. IEEE Conference on Decision and Control, pp. I: 61-66, 2000.

2. E. Altman, T. Jiménez, and D. Kofman. DPS queues with stationary ergodic service times and the performance of TCP in overload, Proc. INFOCOM 2004, 2004.
3. K. Avrachenkov, U. Ayesta, E. Altman, P. Nain, and C. Barakat. The effect of the router buffer size on the TCP performance, Proc. LONIIS International Seminar on Telecommunication Networks and Teletraffic Theory, pp. 116-121, 2002.
4. F. Baccelli and D. Hong. Flow level simulation of large IP networks, Proc. INFOCOM 2003, 2003.
5. S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J.W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level, Proc. ACM SIGCOMM 2001, pp. 111-122, 2001.
6. T. Bonald and L. Massoulié. Impact of fairness on Internet performance, Proc. ACM SIGMETRICS 2001, pp. 82-91, 2001.
7. P. Brown. Resource sharing of TCP connections with different round trip times, Proc. INFOCOM 2000, pp. 1734-1741, 2000.
8. T. Bu and D. Towsley. Fixed point approximation for TCP behavior in an AQM network, Proc. ACM SIGMETRICS 2001, pp. 216-225, 2001.
9. J.W. Cohen. The multitype phase service network with generalized processor sharing, Acta Informatica, Vol. 12, pp. 245-284, 1979.
10. G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many classes, J. ACM, Vol. 27, pp. 519-532, 1980.
11. N. van Foreest, M. Mandjes, and W. Scheinhardt. A versatile model for asymmetric TCP sources, Proc. ITC 18, pp. 631-640, 2003.
12. R. Gibbens, S. Sargood, C. van Eijl, F. Kelly, H. Azmoodeh, R. Macfadyen, N. Macfadyen. Fixed-point models for the end-to-end performance analysis of IP networks, Proc. 13th ITC Specialist Seminar, 2000.
13. Homepage, Network Simulator II, `http://www.isi.edu/nsnam/ns/`
14. F. Kelly. Blocking probabilities in large circuit-switched networks. Adv. Appl. Prob., Vol. 18, pp. 473-505, 1986.
15. F. Kelly. Mathematical modeling of the Internet, Mathematics Unlimited - 2001 and Beyond, pp. 685-702, Springer-Verlag, Berlin, 2001.
16. A. Kherani and A. Kumar. Stochastic models for throughput analysis of randomly arriving elastic flows in the Internet. Proc. INFOCOM 2002, pp. 1014-1023, 2002.
17. P. Kuusela, P. Lassila, J. Virtamo, and P. Key. Modeling RED with idealized TCP sources. Proc. IFIP ATM & IP 2001, 2001.
18. T.V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Trans. Netw., Vol. 5, pp. 336-350, 1997.
19. P. Lassila, H. van den Berg, M. Mandjes, and R. Kooij. An integrated packet/flow model for TCP performance analysis, Proc. ITC 18, pp. 651-660, 2003.
20. L. Massoulié and J.W. Roberts. Arguments in favour of admission control for TCP flows, Proc. ITC 16, pp. 33-44, 1999.
21. M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm, Comp. Comm. Rev., Vol. 27, pp. 67-82, 1997.
22. V. Misra, W. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED, Proc. ACM SIGCOMM 2000, pp. 151-160, 2000.
23. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation, Proc. ACM SIGCOMM 1998, pp. 303-314, 1998.
24. J. Roberts, U. Mocci, and J. Virtamo. *Broadband network teletraffic; Final report of action COST 242,* Springer, Berlin, 1996.

A Multizone Pipelined Cache for IP Routing

Soraya Kasnavi, Paul Berube, Vincent C. Gaudet, and José Nelson Amaral

Dept. of Electrical and Computer Engineering, University of Alberta,
Edmonton, Alberta, T6G 2V4, Canada
{kasnavi, vgaudet}@ece.ualberta.ca
{berube, amaral}@cs.ualberta.ca

Abstract. Caching recently referenced IP addresses and their forwarding information is an effective strategy to increase routing lookup speed. This paper proposes a multizone non-blocking pipelined cache for IP routing lookup that achieves lower miss rates compared to previously reported IP caches. The two-stage pipeline design provides a half-prefix half-full address cache and reduces the cache power consumption. By adopting a very small non-blocking buffer, the cache reduces the effective miss penalty. This cache design takes advantage of storing prefixes but requires smaller table expansions (up to 50% less) compared with prefix caches. Simulation results on real traffic display lower cache miss rate and up to 30% reduction in power consumption.

Keywords: IP lookup, IP Caching, Content Addressable Memory (CAM).

1 Introduction

The sustained increase in Internet traffic over the last decade has necessitated faster and faster backbone networks and a corresponding increase in network processor throughput. A fundamental task in routing IP traffic is finding each packet's destination address and corresponding next hop information in a routing table. Routing tables store routing prefixes, rather than full destination addresses, in order to reduce table size. Multiple prefixes may match a particular destination address. With Classless Inter-Domain Routing (CIDR), routing prefixes may have any length (0 to 32 for IPv4) [15]. In the case that multiple prefixes match an address, the correct lookup result is defined as the longest matching prefix. Consequently, routers must perform Longest Prefix Matching (LPM) when searching the routing table. Since LPM is performed in every router along the packet's path from source to destination, routers require a fast mechanism to perform the lookup in order to maintain high throughput and low latency under load.

An effective strategy to speed up routing lookup is to use a cache to store recent routing results for reuse. The performance of the cache depends on the characteristics of the IP traffic, such as its *temporal* and its *spatial* locality. Greater temporal locality increases the probability that destination addresses are frequently used, and thus increases the utility of a cached address. Spatial locality means referencing addresses in the same numerical range. When prefixes are cached, a single cache entry can cover a large number of destination addresses in a same numerical range. Therefore, the spatial locality in the traffic stream is converted to temporal locality in the cache access stream.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 574–585, 2005.

A cache naturally exploits temporal locality. However, routers manage traffic from a large number of hosts. In some cases, only part of the traffic has high locality. In a router with a single cache low-locality traffic pollutes the cache with low-utility entries. These entries reduce the effectiveness of the cache for all traffic, and may cause thrashing. However, if the cache is split then the cache performance is improved [16, 5]. One portion of a split cache stores addresses or prefixes associated with shorter routing prefixes, and the other portion caches the addresses of prefixes associated with the longer routing prefixes. Such a *multizone* cache prevents the lack of locality in one portion of the traffic from polluting the locality in the rest of the traffic.

This paper proposes a novel multizone pipelined cache (MPC). We study a two zone MPC where one zone stores IP prefixes of 16 or fewer bits in length, while the second zone stores full addresses. This half-prefix half-full IP MPC has a lower miss rate when compared with full address caches. An MPC requires less lookup table expansion and on-chip area than full prefix caches. MPC uses a small buffer to store recent IP addresses that missed the cache. This non-blocking feature allows the cache to resume searching for other IP addresses during a miss. Hit-over-miss effectively reduces the miss penalty. Furthermore, vertical and horizontal pipelining reduces power consumption and increases throughput.

The remainder of this paper proceeds as follows: Section 2 discusses related work. Section 3 describes the cache architecture and cache operation. Consistency issues specific to IP prefix caching are discussed in Section 4. Results from simulation of the cache are presented in Section 5, and we conclude in Section 6.

2 Related Work

Many researchers have addressed the efficiency of routing caches. Some studied existing locality in IP traffic [9, 5]. Others designed more efficient caches for IP routing [6, 13, 16]. In 1988, Feldmeier demonstrated that a routing-table cache could reduce the lookup time in network gateways by 65% [9]. Berube *et al.* designed a method to implement a high density, fully associative CAM-based cache in the Xilinx VirtexE FPGA architecture [2]. Their design is further discussed in Section 3. Chiueh *et al.* designed a CPU style IP caching scheme and demonstrated that general-purpose processors can serve as a powerful platform for high performance IP routing [6]. Since the data streams presented to the network processors have very different characteristics than the streams accessed by general-purpose CPUs, the cache design must be considerably different and the cache coverage must be improved to achieve acceptable performance [4]. A network cache should cache address ranges rather than individual addresses and its blocks should be small. Liu proposes IP Prefix Caching to achieve lower miss rates due to higher locality in prefixes compared to individual addresses [13]. Prefix caching is very efficient due to increased spatial locality but the lookup table of prefix caches should be transformed to assure correct cache results [13, 16]. Section 4 presents this transformation. Cache miss rate can improve when the cache is divided into different zones dedicated to different lookup prefix lengths. Chvets and MacGregor studied a *multizone* cache [5]. They simulated a two-zone full address cache where IP addresses are stored in each zone according to their lookup result prefix lengths. Their new cache design shows miss ratios approximately one-half those of conventional caches.

IP caches have very large miss penalties because a miss requires a rather slow main table lookup. Complicated lookup techniques can be applied to the main table to increase the lookup speed. However, these techniques dramatically increase the table updating delays. Thus, improving the cache miss ratio could compensate for the large cache miss penalty and allow a simple main lookup table to provide fast table updates. Non-blocking general purpose processor caches hide memory latency by overlapping the processor computations with memory data accesses [3]. Special registers are used to hold information about each cache miss. The processor can then overlap the service of a cache read miss with the execution of the subsequent instructions [8, 11]. Bhuyan *et al.* used execution-driven simulation to study the impact of instruction level parallelism (ILP) and cache architectures on the performance of routers [12]. They observed up to 37% improvement for their traces due to multiple issues, out of order executions and non-blocking loads.

This paper proposes MPC, a multizone, non-blocking, pipelined cache. MPC uses prefix caching in multiple zone caches to improve cache miss ratio. MPC adopts a non-blocking buffer to reduce the effective cache miss penalty. A pipelined design implements a novel search and reduces power consumption. The details of the MPC architecture and features are further described in Section 3.

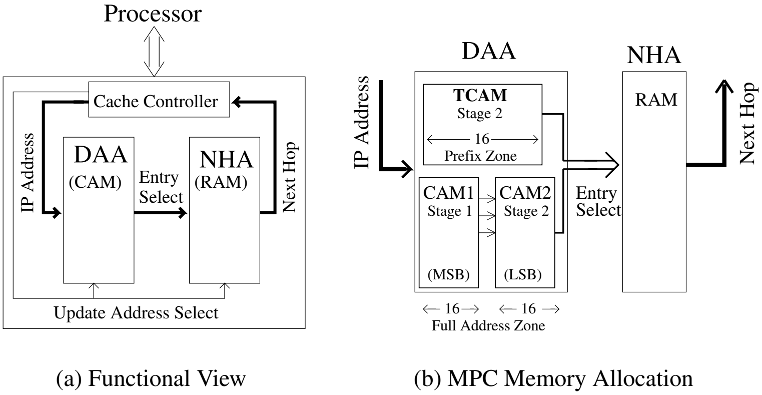
3 Architecture

Figure 1(a) presents a functional block diagram of a two zone MPC. MPC is based on the previous work of Berube *et al.* [2]. IP addresses or prefixes are stored in a *Destination Address Array* (DAA). Next hop information is stored in the *Next Hop Array* (NHA). The DAA and NHA are co-indexed, with one entry in the NHA corresponding to a single entry in the DAA.

MPC searches all entries of the DAA for an IP address. If the MPC finds the address in the DAA, a cache hit occurs, and the corresponding next hop from the NHA is returned to the processor. If no entry in the DAA matches the IP address, a cache miss occurs. In this case, a lookup in the full routing table is performed and the cache is updated with the new destination address/next hop pair.

Figure 1(b) presents a structural description of MPC. The DAA is divided into two parts horizontally. The two parts form the two zones of the cache, and have independent sizes. The upper *Prefix Zone* stores *short* IP prefixes, which are 16 or fewer bits long. The lower *Full Address Zone* stores full 32-bit IPv4 destination addresses. The *Full Address Zone* is further divided vertically, with each entry split in half. The most significant 16 bits of the address are stored in `CAM1`, while the least significant 16 bits are stored in `CAM2`.

The NHA is implemented using standard SRAM technology. The DAA is implemented using a combination of Content Addressable Memory (CAM) and Ternary CAM (TCAM) cells. A CAM is a fully associative binary memory capable of matching a specific pattern of data (a key) against all its entries in parallel. A CAM is used to implement the two halves of the *Full Address Zone*. A TCAM is used to implement the



(a) Functional View

(b) MPC Memory Allocation

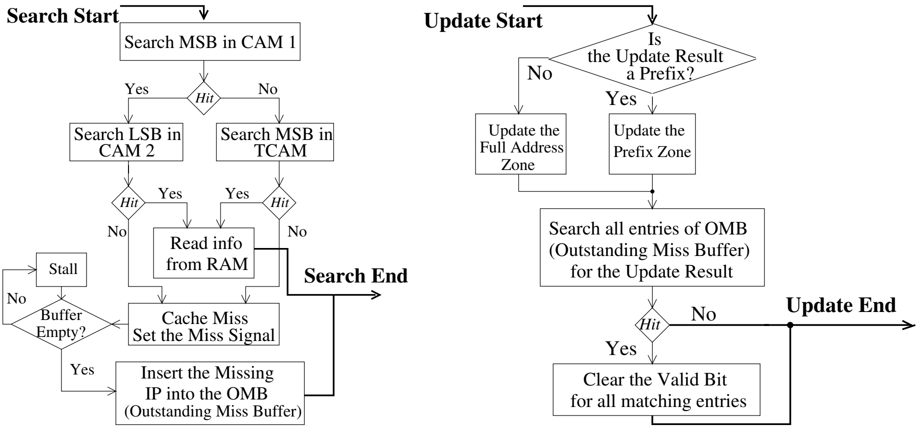
Fig. 1. General Description of the Cache Architecture

Prefix Zone of the cache because a TCAM can store *don't care* states in addition to 0s and 1s. TCAM bits set to the *don't care* state will match both 0s and 1s in the key. Thus, a TCAM is well suited to store IP prefixes.

3.1 Cache Functionality

Breaking the DAA into three pieces allows cache lookups to be pipelined. The pipeline has three stages: (1) a lookup in `CAM1`; (2) a lookup in either `CAM2` or the *Prefix Zone*, as required by the results of the first stage; (3) an access to the *NHA RAM* (on a hit) to return the lookup result (forwarding information or cache miss indication). In stage 1 the most significant 16 bits of the address are applied to `CAM1`. If there are any matches in `CAM1`, in stage 2 the corresponding entries of `CAM2` are searched with the 16 least significant bits of the address to complete the full-IP match in the *Full Address Zone*. Otherwise, stage 2 applies the 16 most significant bits of the address to the prefixes cached in the *Prefix Zone*. If there is a match in either the *Full Address Zone* or the *Prefix Zone*, stage 3 accesses the RAM location corresponding to the matching entry, and returns the next hop data as the lookup result. Note that if the IP address hits CAM1, the prefix zone is not searched at all. The correctness of this lookup scheme requires that each IP address either hit the cache in the full address zone or in the prefix zone, but never in both. The routing table transformations required to ensure correct cache results are described in Section 4.

When there is no match either in the Full Address Zone or in the Prefix Zone, a cache miss is reported. A routing table search returns the routing information that is then stored in the MPC. The time required to complete this search and store the value in the cache is known as *miss penalty*. Servicing a miss is time consuming because of the slow main memory accesses to the routing table. MPC stores recent misses in an *Outstanding Miss Buffer* (OMB) until the processor returns the lookup results (see Section 3.2). Figure 2(a) depicts the pipeline flow diagram for a cache search. The diagram for an update is in Figure 2(b).


Search Start
Search MSB in CAM 1
Yes
Hit
No
Search LSB in CAM 2
Search MSB in TCAM
Hit
Yes
Yes
Hit
No
Read info from RAM
No
Stall
Search End
No
Buffer Empty?
Cache Miss Set the Miss Signal
Yes
Insert the Missing IP into the OMB (Outstanding Miss Buffer)
Update Start
Is the Update Result a Prefix?
No
Yes
Update the Full Address Zone
Update the Prefix Zone
Search all entries of OMB (Outstanding Miss Buffer) for the Update Result
Hit
No
Update End
Yes
Clear the Valid Bit for all matching entries

(a) Cache Search (b) Cache Update

Fig. 2. Flow Diagram of the Cache performance

3.2 Outstanding Miss Buffer

MPC uses the OMB to store recent misses until the processor returns their lookup results. Without OMB, MPC would need to stall while each cache miss is serviced. Blocking hinders cache throughput because further cache searches cannot proceed until the lookup is performed and the cache updated, even if pending request would hit the cache.

When a miss occurs in the non-blocking MPC, the address is stored in the OMB and the cache continues performing lookups. If subsequent IP addresses hit the cache while a miss is being serviced, a *hit under miss* occurs. A *miss under miss* (secondary miss) occurs when a subsequent IP address also misses the cache. Secondary misses are stored in the OMB until the buffer is full, at which point MPC blocks and the processor stalls until misses are serviced and removed from OMB. An example of MPC functionality with a two-entry OMB is given in Figure 3. In this example, IP2, IP4 and IP6 are cache misses. The MPC is able to search for IP3 and IP5 and forward their corresponding information to the processor while the main table lookup for IP2 is in progress. The MPC stalls after searching for IP6 because OMB is full. No new IP can be searched until IP2 is serviced and removed from OMB to make room for IP6.

When the lookup result of a pending IP address in the OMB comes back from the main memory lookup, the MPC updates either the prefix zone or the full address zone with the corresponding information according to the MPC replacement policy. An expansion of the lookup table ensures that the result is either a short prefix that updates the prefix zone, or a full address with 32 bits that updates the full address zone (see Section 4).

MPC requires a pipeline stall to update the data stage by stage. After an update is complete, the missing IP is removed from OMB. However, other pending IP addresses in the OMB might be identical to the recently updated IP address. Additionally, an

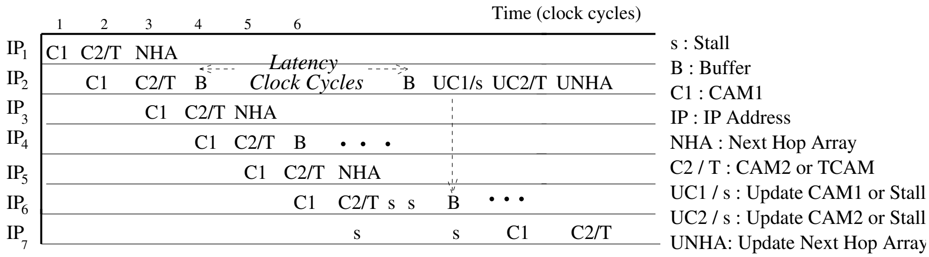
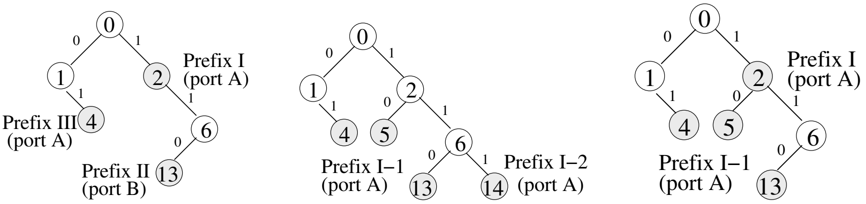


Fig. 3. Pipeline Diagram of the Cache

update result might be a prefix covering multiple pending IP addresses in the OMB. To ensure that the same lookup result is not written into the cache multiple times, we implement the OMB as a 33-bit CAM. Each OMB entry stores a 32 bit address and a *valid bit*. Only valid entries require a software lookup and cache update. After each update, an associative search of OMB identifies all matching entries. If the lookup result is a prefix, its *don't care* bits are externally masked to ensure that they match with the data in OMB. The valid bits of all matching entries are cleared. A second search for those matching OMB entries will now hit the cache, and provide the processor with the next hop information. The flow diagram for an MPC update is shown in Figure 2(b).

4 Lookup Table Transformation

Caching prefixes reduces the cache miss rate because a stored prefix can cover a large range of the address space. However, routers are required to provide Longest Matching Prefix routing. If multiple prefixes match an address, a situation may arise where the longest matching prefix is not present in the cache, but a shorter matching prefix is in the cache. This short matching prefix will produce a cache hit, leading to an incorrect routing decision. Figure 4(a) depicts an example of three prefixes of a routing table organized as a *trie*. A trie presentation of a lookup table is a tree-based scheme where the root of the trie corresponds to the most significant bit of the address. Branching right indicates that a bit is 1, while branching left indicates that a bit is 0. A complete trie enumerates every possible address. In order to reduce the space requirement of the trie, only the nodes required to form a path to each prefix are stored. Nodes are sequentially numbered from top to bottom and from left to right. Gray nodes represent prefixes stored in the lookup table. The prefix in node 2 (prefix I) is said to encompasses the prefix in node 13 (prefix II), because node 2 is on the path from the root to node 13. Encompassing prefixes are responsible for situations where a prefix cache may produce incorrect routing decisions. If Prefix I is cached, it will match with IP addresses whose longest matching prefix in the trie is Prefix II. An IP address matching Prefix II could then be incorrectly matched by Prefix I and forwarded to port A. Therefore, encompassing prefixes are *non-cacheable*. In this example, the prefix at node 4 (prefix III) is *cacheable* because it does not encompass any other prefix.



(a) Original trie (no expansion)

(b) Full trie expansion

(c) Partial trie expansion in [13]

Fig. 4. Trie presentation of a small lookup table

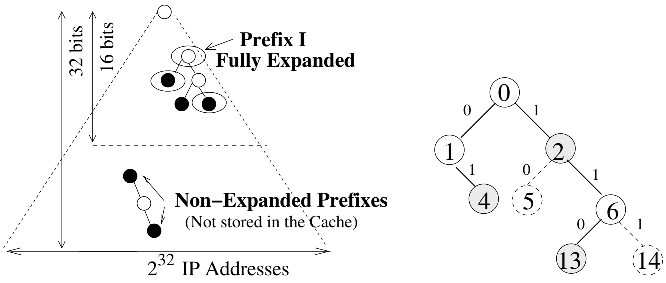
4.1 Prefix Caching and Table Expansion

One general solution to ensure correct routing cache results is to cache only *cacheable* prefixes and cache full IP addresses when lookup results are *non-cacheable* prefixes [13, 16]. This solution requires no lookup table transformation, but the cache miss rate will be higher because of the reduced coverage afforded by the full addresses placed in the cache. Moreover, the lookup scheme must decide if a prefix is cacheable or not. Other solutions expand the trie to ensure that all prefixes are leaves of the trie, and thus *cacheable*. Figure 4(b) shows a complete expansion of the trie for our example. Prefix I is expanded by appending ”0” to form prefix I-1 in node 5, and by appending ”11” to form prefix I-2 in node 14 and the forwarding information (port A) is copied in both nodes 5 and 14. Observe that the number of valid prefixes increases and the lookup table gets larger. Liu reports up to an 118% increase in table size [13]. Routing table expansion is unfavorable due to memory area limitations, power consumption and cost. On the other hand, table expansion pushes prefixes lower in the trie, increasing the search time for Software searches. Also, updates in a fully expanded table are challenging, since an expanded prefix no longer exists and the update must find the prefixes created by expansion. Liu presents a partial prefix expansion to reduce the number of prefixes in the lookup table for a prefix cache [13]. Figure 4(c) depicts the partial expansion in [13] where non-cacheable prefixes are expanded only to their first level of expansion. Prefix I-1 is added to the trie at node 5 with same forwarding information as prefix I. Observe that routing table growth is less than with full expansion, but the lookup scheme must still decide if a prefix is *cacheable* or not.

4.2 Table Expansion in MPC

Our MPC solution for table expansion is illustrated in Figure 5(a). MPC fully expands the routing table for short prefixes, pushing them down the tree until they either become leaf nodes, or become 17 bits long. All short prefixes are thus cacheable in the Prefix Zone of the MPC. Any destination address matching a long prefix in the lookup table is stored in full in the Full Address Zone of the MPC. This table transformation has the following advantages:

1. Routing table expansion is limited to those prefixes that provide the greatest coverage of the IP address space.



(a) The Proposed Partial Table Expansion)

(b) Expansion Free Transformation

Fig. 5. Table Transformation

2. At most one prefix stored in the Prefix Zone can match a destination address, since all short prefixes are cacheable.
3. Since the short prefixes in the Prefix Zone are cacheable, no address can hit both the Prefix Zone and the Full Address Zone. Therefore, any address that could hit the Prefix Zone is guaranteed to miss the Full Address Zone.
4. A length check is sufficient to determine if a prefix is cacheable.

Note that points 2 and 3 guarantee that there cannot be multiple hits for a search in the cache, which simplifies cache design. Also, point 3 enables our power-saving pipelined search.

4.3 Expansion-Free Software Lookups

Lookups may be implemented in hardware or software. A software lookup walks down the trie to find the longest matching prefix for an IP address [15]. The longest matching prefix is the last prefix encountered in the trie. Some of these longest matching prefixes might not be cacheable in a non-expanded table. We propose a new *Expansion-Free* (EF) method to generate cacheable prefixes using a simple and inexpensive mechanism during a software lookup. EF forwards the generated cacheable prefixes to the cache but does not store them in the lookup table, thus eliminating problems associated with table expansion. Since the original submission of this paper, we learnt that Akhbarizadeh *et al.* independently developed a similar method [1]. Figure 5(b) illustrates the EF method on the example of Figure 4. Let n be the last node visited during a traversal of the trie for an IP address, and let p be the last node visited containing a prefix during the traversal. EF has three rules.

1. If $p = n$ is a leaf node in the trie, then the prefix in p is cacheable and can be forwarded to the prefix cache. For example, for IP addresses covered by node 4, $p = n = 4$. Thus the prefix in node 4 is cacheable and is forwarded to the cache as the lookup result.

2. If $p = n$ is not a leaf node in the trie, p is not cacheable. In Figure 5(b), assume an IP address matches node 5, $n = p = 2$. A cacheable prefix can be produced by adding 0 (the next bit in the address) to the path followed to encounter p. This generated cacheable prefix is forwarded to the cache.
3. If $p \neq n$, the prefix in p is not cacheable. In Figure 5(b), assume an IP address matches node 14 in the trie. For this IP address, $p = 2$ and $n = 6$. A cacheable prefix is produced by adding a 1 (the next bit in the address) to the path traversed to find n. This generated cacheable prefix is forwarded to the cache.

5 Performance Evaluation

To evaluate the MPC design we build a high level architectural simulator and run it with real IP traces and lookup tables of three distributing (neither core nor edge) routers of local service providers. The characteristics of the data used for simulations are given in Table 1.

Table 1. Trace Characteristics

	ISP1	ISP2	ISP3
Trace Length (Packets)	99117	98948	98142
Routing Table Size (Prefixes)	10219	10219	6355

To evaluate the performance improvements achieved by MPC, we compare the miss rates of MPC with a full address IP cache and a full prefix cache. For a fair comparison, the IP cache is simulated as a two-zone two-stage pipelined cache with equal sized zones. This architecture caches full IP addresses and is implemented in a 32 bit binary CAM. The prefix cache is a 32-bit Ternary-CAM that store prefixes, using a fully expanded version of the real lookup table (LUT). The miss rates are given in Table 2.

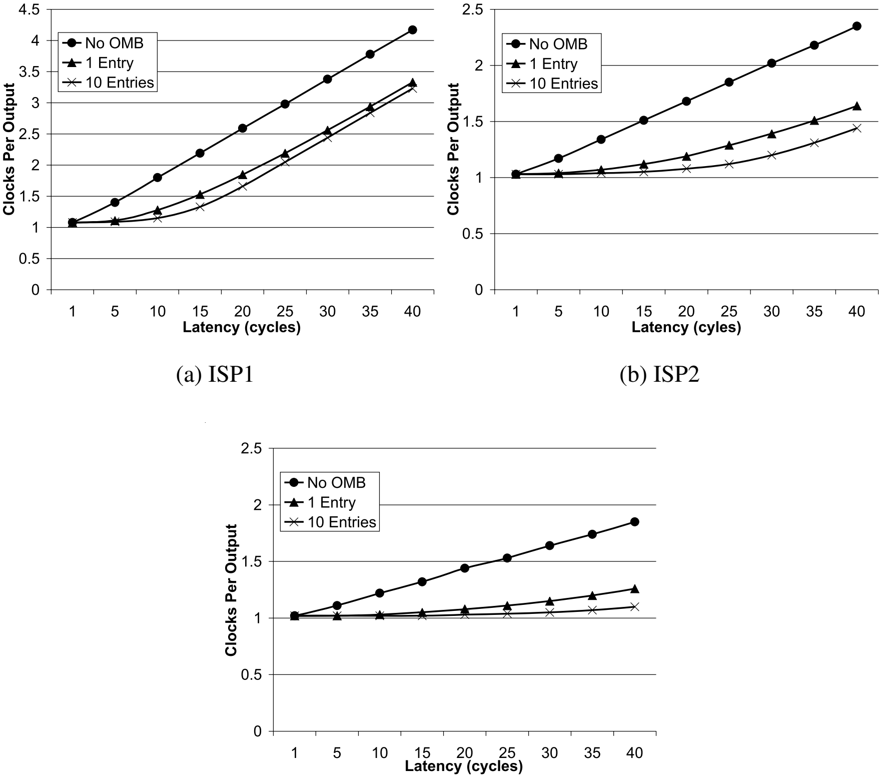
Table 2. Miss Rates (%) vs. Cache Sizes (No. of Entries) for Three Traces

	ISP1			ISP2			ISP3		
Entries	**IP**	**Proposed**	**Prefix**	**IP**	**Proposed**	**Prefix**	**IP**	**Proposed**	**Prefix**
512	22.7	15.5	7.4	10.8	6.2	2.9	3.6	3.0	0.5
1024	15.4	7.9	2.5	7.2	3.3	1.3	2.2	2.0	0.5
2048	10.5	3.7	1.4	4.9	2.0	1.2	1.9	1.6	0.5

Clearly, the prefix cache outperforms the two other caches with the same number of entries. However, the prefix cache must be implemented in a 32-bit TCAM. Since the area required for a TCAM cell is almost twice the area of a CAM cell, MPC and the IP Cache use half the area of a prefix cache with the same number of entries. To compare the performance of caches with the same storage area, MPC and the IP cache should be compared to a prefix cache with half as many entries. The simulation results

Table 3. Number of Prefixes after Table Expansion

	ISP1		ISP2		ISP3	
	Entries	% Larger	Entries	% Larger	Entries	% Larger
Original Table	10219	–	10219	–	6355	–
Prefix Cache	30620	199	30620	199	7313	15
MPC	17485	71	17485	71	6469	2



(a) ISP1

(b) ISP2

(c) ISP3

Fig. 6. CPO vs. Latency for 1K-Entry (equally sized zones) MPC

indicate that for equal cache size (storage area), the performance of MPC is almost as good as the prefix cache. Moreover, the prefix cache requires full LUT expansion while MPC requires a partial expansion (as described in Section 4). Table 3 compares the total number of prefixes in the LUT after the expansion for a prefix cache and MPC.

MPC uses a small buffer (OMB) to hide the miss penalty. The miss penalty is modeled in our simulator by a *latency* parameter. A cache that has no buffer to store recent misses has to stall at each miss and wait until the update result is returned to the cache. To evaluate the impact of the miss penalty we measure a metric called CPO (Clock Per

Output) that reports the average number of clock cycles necessary to provide the Next Hop Information for an IP address. Figure 6 depicts CPO versus Latency for MPC with no OMB, OMB with a single entry, and OMB with 10 entries. As expected, CPO increases linearly with latency for a cache with no buffer. For small latencies, in an MPC with a single entry OMB, the CPO is almost independent of the latency. For larger values of latency, CPO again increases linearly, but remains less than without the OMB.

Table 4. CAM1 Hit Rates

# Entries	ISP1 %	ISP2 %	ISP3 %
512	62	64	74
1024	63	65	74
2048	63	65	74

5.1 Power Savings

In a CAM-based device, power consumption is an important constraint that is addressed by many designs [7, 14]. The power consumption in a CAM-based device can be separated into three components: Evaluation Power (Search power), Input Power and Clocking Power [10]. All these sources are linearly dependent on the number of entries searched. If 50% of the time, only half of the entries of the cache are searched, the effective number of entries during each search operation is reduced to 75% of the physical number of entries. Thus 25% power is saved.

MPC divides the cache entries into two zones: the TCAM prefix zone and the CAM full address zone. For the study of power consumption, we assume that each zone contains half of the cache entries. The prefix zone is searched only if the address misses CAM1 of the full address zone. Our simulation results, presented in Table 4, indicate that almost 60% of the IP addresses hit CAM1, eliminating the need to search the TCAM. This results in a 30% reduction in the effective number of entries searched in the cache, and a corresponding 30% power savings compared to caches that search all entries.

6 Conclusion

We have proposed MPC, a non-blocking, multizone-pipelined-cache that dedicates different zones to different lookup prefix lengths. The *Prefix Zone* is able to store and search prefixes with 16-bits or less. The *Full Address Zone* stores and searches for full IP addresses whose lookup prefixes are more than 16 bits long. Prefix caching increases the cache coverage while a relatively small table expansion is required. EF method, proposed in this paper, completely eliminates table expansion for software lookups. Also MPC potentially can achieve higher throughput and low power consumption due to pipelining. The effective miss penalty is also reduced by using the non-blocking buffer to let the cache search for new IPs while waiting for the lookup results of cache misses.

References

1. M. J. Akhbarizadeh and M. Nourani. Efficient prefix cache for network processors. In *12th Annual IEEE Symposium on High Performance Interconnects*, pages 41–46, Aug 2004.
2. P. Berube, A. Zinyk, J.N. Amaral, and M. MacGregor. The bank nth chance replacement policy for FPGA-based CAMs. In *13th International Conference on Field Programmable Logic and Applications (FPL)*, Lisbon, Portugal, September 2003.
3. T. Chen and J. Baer. Reducing memory latency via non-blocking and prefetching caches. In *5th Int. Conf. Architectural Support for Programming Languages and Operating Systems*, pages 51–61, Oct 1992.
4. Tzi-Cker Chiueh and Prashant Pradhan. Cache memory design for network processors. In *Sixth International Symposium on High-Performance Computer Architecture*, pages 409–419, Toulouse, France, January 2000.
5. I.L. Chvets and M. MacGregor. Multi-zone caches for accelerating IP routing table lookups. In *Merging Optical and IP Technologies Workshop on High Performance Switching and Routing*, pages 121–126, May 2002.
6. Tzi cker Chiueh and Prashant Pradhan. High performance IP routing table lookup using CPU caching. In *IEEE INFOCOM (3)*, pages 1421–1428, 1999.
7. A. Efthymiou and J.D. Garside. A CAM with mixed serial-parallel comparison for use in low energy caches. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, 12:325–329, March 2004.
8. K. I. Farkas and N. P. Jouppi. Complexity/performance tradeoffs with non-blocking loads. In *21st Int. Symposium on Computer Architecture*, pages 211–222, 1994.
9. D.C. Feldmeier. Improving gateway performance with a routing-table cache. In *IEEE INFOCOM 88*, pages 298–307, March 1988.
10. C. Jen; H. Hsiao, D. Wang. Power modeling and low-power design of content addressable memories. In *IEEE Int. Symposium on Circuits and Systems*, pages 926–929, May. 2001.
11. D. Kroft. Lookup free instruction fetch/prefetch cache organization. In *8th Int. Symposium on Computer Architecture*, pages 81–87, May 1981.
12. H. Wang L. Bhuyan. Execution-driven simulation of IP router architectures. In *IEEE Int. Symposium on Network Computing and Applications*, pages 145–155, Oct. 2001.
13. H. Liu. Routing prefix caching in network processor design. In *Tenth International Conference on Computer Communications and Networks*, Oct 2001.
14. K. Pagiamtzis and A. Sheikholeslami. Pipelined match-lines and hierarchical search-lines for low-power content addressable memories. In *IEEE Custom Integrated Circuit Conference*, September 2003.
15. M.A. Ruiz-Sanchez, E.W. Biersack, and W. Dabbous. Survey and taxonomy of IP address lookup algorithms. *IEEE Network*, 15:8–23, April 2001.
16. W.L. Shyu, C.S. Wu, and T.C. Hou. Multilevel aligned IP prefix caching based on singleton information. In *GLOBECOM 02*, volume 3, page 2345 2349, Nov 2002.

Aggregated Aggressiveness Control on Groups of TCP Flows

Soohyun Cho and Riccardo Bettati

Computer Science Department, Texas A&M University,
College Station, TX 77843 USA
{s0c6496, bettati}@cs.tamu.edu

Abstract. The use of multiple concurrent parallel TCP flows is an easy way to achieve higher speed reliable data transfers. However, parallel TCP flows are inherently unfair with respect to single TCP flows. We suggest a new scheme called TCP-P, which controls aggressiveness of a group of parallel TCP flows by regulating their total aggressiveness (or unfairness) to be comparable to a single TCP flow, or any multiple thereof. TCP-P makes a group of N parallel TCP flows appear to other flows like k separable TCP flows - i.e., have strength k - through appropriate manipulations of increase and decrease behavior of the congestion windows of the TCP flows in the group. We implemented our scheme as part of Linux and experimental results show that the proposed scheme effectively controls aggressiveness of parallel TCP flows.

1 Introduction

A widely used scheme to work around the limitations of TCP over high delay-bandwidth product connections is to use multiple parallel TCP connections. The use of parallel TCP flows has several benefits compared to a single TCP flow [1]. If an end-host opens N parallel TCP flows to the same destination, its congestion window recovery and increase are N times faster than a single TCP flow [2]. As a result, the achievable throughput of parallel TCP flows is significantly bigger than that of a single TCP flow given the same packet loss probability. Unfortunately, this increase comes at the expense of the throughput experienced by other, single TCP flows, as sender nodes who open multiple parallel TCP flows will consume unfairly more bandwidth when they compete for the same bottleneck links.

With the increased venues for bundling of TCP flows (e.g., overlay networks with TCP splicing [3], large servers with topological aggregation of service delivery, dedicated connections between supercomputers or campuses, etc.,) flexible schemes are needed for the *controllable* aggregation of large numbers of parallel TCP flows. Naively limiting the number of parallel connections that applications in a node can open concurrently is not appropriate in many situations, as it violates the separation of application design from network resource allocation: Making the number of available connections visible to the application unduly

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 586–597, 2005.

burdens the application design. On the other hand, hiding the varying numbers of connection endpoints from the application through tunneling or multiplexing schemes typically is costly. Also, statically limiting the maximum aggregate sending rate of parallel TCP flows from sender nodes may leave network resources under-utilized because it disables TCP's available bandwidth probing ability beyond the given sending rate. It is therefore preferable to allow for TCP flows to aggregate, but do so in a controlled way.

Methods to control aggregation of TCP flows must have the following capabilities:

- Transparency to applications (management of connections and expected dynamics of data transmission should maintain TCP characteristics,)
- Compatibility with existing TCP implementations ("TCP-friendliness",)
- Controllability and flexibility of the service (the "control knob" offered by the mechanism should be intuitive and have measurable effect on behavior,)
- Effective use of available bandwidth (the mechanism should not prevent TCP from quickly making use of available bandwidth,)
- Flexible deployability (the mechanism should be deployable in single-sender (server), or multi-sender (overlay) scenarios.)

In this paper we propose aggregate *strength* as means to control the fairness of parallel TCP flows: The aggressiveness or unfairness of parallel TCP flows is appropriately controlled to not exceed that of a configurable number of single TCP flows, regardless of the number of TCP connections. With the term fairness (unfairness) we mean how fairly (unfairly) a group of parallel TCP flows from a node share network resources such as bandwidth with TCP flows from other nodes. By setting the aggregate strength of a group to some value k, the group of parallel TCP flows in a node behaves as if there were a group of k parallel TCP flows regardless of the number of parallel flows applications in the node open. By doing so we provide a flexible aggregate control of parallel TCP flows while keeping the ability of parallel TCP flows to effectively utilize available bandwidth.

We implement strength control within TCP-P, which is an extension to TCP. TCP-P controls the aggressiveness of a group of N parallel TCP flows from a node against single TCP flows from other nodes by controlling the strength of the group of flows. The "strength" in this context is a scalar value k of the TCP group and describes how big (in terms of number of flows) the group is perceived by other TCP flows from other nodes sharing network resources with the group. We will show in the following how this parameter provides a simple and intuitive means to control aggressiveness of parallel TCP flows.

There have been several efforts to improve TCP performance using parallel flows while constraining the unfairness of parallel TCP flows comparable to that of a *single* TCP flow. The Congestion Management (CM) architecture [4], Fractional/Combined TCP flows [5] and COCOON [6] are some examples. In contrast to these schemes, MulTCP [7] was proposed to claim k times more bandwidth for a single TCP connection. A MulTCP flow with parameter k increases and decreases its congestion window size as if there were k multiple TCP flows. However, as far as we know, there has been no scheme to controllably constrain the aggressiveness of parallel TCP flows.

The remainder of this paper is organized as follows: Section 2 presents the methodology we used to control parallel TCP flows' total strength. Section 3 describes our implementation in the Linux kernel. Section 4 presents experimental results that demonstrate how this implementation effectively controls the aggressiveness of parallel TCP flows. Section 5 concludes this paper.

2 Aggregate Control

Aggregate control of parallel TCP flows is achieved through modifications to TCP's congestion window increase and decrease behavior. During the increase phase, a normal TCP has two modes: exponential increase during slow-start, linear increase during congestion avoidance. Within the decrease phase, normal TCP responds to congestion events, such as three duplicate acknowledgement packets (ACKs,) by halving its congestion window size. With a given strength parameter k, we want to match the total amount of increase and decrease of congestion windows of a group of N parallel TCP-P flows to those of k single TCP flows.

We denote the amount of *increase* of congestion window of a single TCP flow i in increase phase as Δ_i^+ and the amount of *decrease* in decrease phase as Δ_i^-, respectively. Let the amount of increase and decrease of a TCP-P flow j in a group of size N be Δ_j^{p+} and Δ_j^{p-} respectively. To make N parallel TCP-P flows become like k single TCP flows, we need to make sure that $\sum_{i=1}^{k} \Delta_i^+ = \sum_{j=1}^{N} \Delta_j^{p+}$ for the given number of non-duplicate ACKs, and $\sum_{i=1}^{k} \Delta_i^- = \sum_{j=1}^{N} \Delta_j^{p-}$ for a congestion event.

2.1 Controlling Increase

In slow-start mode,TCP increases its congestion window by one pe rnon-duplicate ACK until it detects congestion events or the congestion window size reaches its slow-start threshold value. When a TCP is in slow-start mode, the congestion window size of a TCP, W, after a non-duplicate ACK arriving at time t is shown in the following equation:

$$W(t+) = W(t) + 1. \tag{1}$$

When all TCP flows in a group of N unmodified parallel TCP flows are in slow-start mode, the total congestion window increase will be N times faster that a single TCP flow. For the same group size of TCP-P flows to have strength k, the congestion window of each TCP-P flow, W_j, should increase by $\frac{k}{N}$ per non-duplicate ACK as shown in the following equation:

$$W_j(t+) = W_j(t) + \frac{k}{N}. \tag{2}$$

In congestion avoidance mode, we want to make the aggregate congestion window size increase of N parallel TCP-P flows with strength k be equal to that of k

TCP flows for the same amount of non-duplicate ACKs. We describe the special case of $k = 1$ first, and generalize it later. Let $W(t)$ be the congestion window size of a single TCP at time t, and we assume the sum of congestion window size of each TCP-P flow in the group is equal to $W(t)$, i.e., $\sum_{j=1}^{N} W_j(t) = W(t)$. The amount of congestion window increase of a single TCP per non-duplicate ACK in this mode is $\frac{1}{W(t)}$ as shown in the following equation:

$$W(t+) = W(t) + \frac{1}{W(t)}. \tag{3}$$

If each TCP-P flow in a group size N increases its congestion window by one, the total increase will be N. For a single TCP flow to increase its congestion window size W by N, it needs $W+(W+1)+(W+2)+\cdots+(W+N-1) = \sum_{i=0}^{N-1}(W+i)$ non-duplicate ACKs. Hence, to match the congestion window increase speed of N parallel TCP-P flows to that of a single TCP flow, we should require the group of TCP-P flows with size N to receive $\sum_{i=0}^{N-1}(W+i)$ non-duplicate ACKs before each TCP-P flow in the group increases its congestion window by one.

To ensure fairness among TCP-P flows within the group, we evenly distribute the total amount of non-duplicate ACKs required for a group to each TCP-P flow in the group, so that each TCP-P flow in a group needs to receive $\frac{\sum_{i=0}^{N-1}(W+i)}{N}$ non-duplicate ACKs before it can increase its window size by one. In this way, with the same amount of non-duplicate ACKs, i.e., $\sum_{i=0}^{N-1}(W+i)$, the N parallel TCP-P flows will increase their total congestion window size by the same amount, N, just as the single TCP flow.

For the case of $k > 1$, we generalize the previous case: parallel TCP-P flows need to increase their total window size of the group by k after the group received $\sum_{i=0}^{N-1}(k\overline{W} + i)$ non-duplicate ACKs. Here, $\overline{W}$ is the average of the congestion window sizes of k TCP flows, and we assume that $\sum_{i=1}^{k} W_i = k\overline{W} = \sum_{j=1}^{N} W_j$, i.e., the sum of congestion window W_i of k single TCP flows is equal to the sum of congestion window W_j of N parallel TCP-P flows at time t. We use the fact that the increase of the total congestion window size of k parallel TCP flows with a given number of non-duplicate ACKs is k times larger than the increase of the congestion window of a single TCP flow with the same window size (i.e., $k\overline{W}$). For this, each TCP-P flow in a parallel TCP-P group of size N should increase its congestion window by one after receiving the following amount of non-duplicate ACKs:

$$\frac{\sum_{i=0}^{N-1}(\sum_{j=1}^{N} W_j + i)}{k * N}. \tag{4}$$

As a result, the increase behavior of a group of N TCP flows can be made to closely reflect that of k TCP flows.

2.2 Controlling Decrease

A single TCP flow reduces its congestion window size W by half when it detects a congestion event, such as three duplicate ACKs at time t:

$$W(t+) = \frac{W(t)}{2}. \tag{5}$$

In unmodified parallel TCP flows, only single TCP flow in the group halves the congestion window size for each congestion event to the group. This behavior primarily contributes to the observed throughput advantage (and the unfairness) of parallel TCP flows over a single flow. In contrast, we let each TCP-P flow in a group responds to its own congestion event by reducing its own congestion window. In addition, TCP-P adjusts congestion windows of *other* TCP-P flows in the group as well based on the group size N and strength parameter k.

For $k = 1$, we halve *all* parallel TCP-P flows' congestion window sizes whenever *any* member flow detects a congestion event. For $k > 1$, we let the total congestion window size after a congestion event be $\frac{2k-1}{2k}$ of the previous total congestion window size of N parallel TCP-P flows. Hence, for a group of N TCP-P flows with strength k, the total amount of congestion window decreases according to the following equation:

$$\sum_{j=1}^{N} W_j(t+) = \sum_{j=1}^{N} W_j(t) * (\frac{2k-1}{2k}). \tag{6}$$

For $k = N$, N parallel TCP-P flows becomes unmodified N parallel TCP, and the total decrease amount of N TCP-P flows become $\frac{2N-1}{2N}$ of the previous total window size. This is the same as that of unmodified parallel TCP flows' [2].

2.3 Avoiding Unnecessary Decreases

Since packet drops in the network are typically bursty [8], multiple TCP flows in a parallel TCP group may simultaneously experience packet losses. If bursty packet drops occur, they may result in too much congestion window reduction to parallel TCP-P flows: Every TCP-P flow that experiences a congestion event might in turn trigger a congestion window reduction in other flows, which already may have responded to the congestion event. This results in a congestion response cascade. In traditional TCP a flow responds to congestion events only once within a congestion window, regardless of the number of lost packets. In comparison, TCP-P flows may end up with a lower throughput than that of a single TCP flow.

To avoid this unnecessary reduction of congestion windows, a TCP-P flow skips adjusting congestion windows of all member TCP-P flows if the elapsed time since its last adjustment by other flows is less than the minimum of the moving average of its round-trip times[1] (i.e., minimum *srtt*.) In doing so, we

[1] This is also called *smoothed* round-trip time, $srtt(t+) = (1-w) * srtt(t) + w * rtt(t)$ where $rtt(t)$ is round-trip time at time t and $w = \frac{1}{8}$.

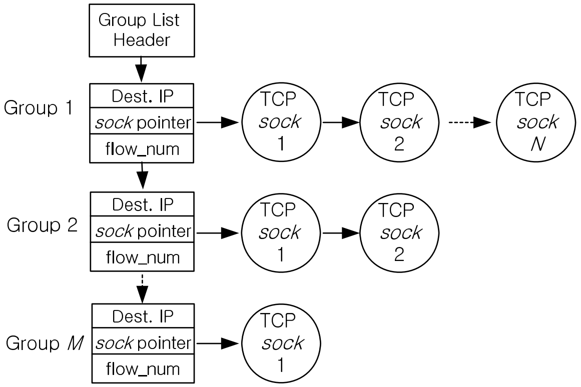


Fig. 1. Manage groups and flows

assume that the length of packet drop bursts does not typically last longer than the minimum of the moving average of round-trip times.

3 Implementation Issues

We implemented TCP-P scheme on Redhat Linux 9.0 kernel 2.4.20-8. The default behavior of Linux TCP implementation is based on TCP-Sack [9], time-stamping on each packet, and Quick-ACK [10]. Also, Linux uses the packet as the unit of congestion window size, unlike BSD, which uses bytes.

3.1 Structure

Whenever a TCP connection is established, the system kernel looks up the *group list* using the destination IP address as a key to know whether other flows already exist to the same destination[2]. If no such group exists, a new group entry is added in the list and the connection is registered as a member of the group using its `sock` structure pointer. Otherwise, the connection is added as a member to the group. When a connection closes, the connection is removed from the list of members of the group. If the group has no more members it is also deleted.

Fig. 1 illustrates how the Linux data structures are extended to manage TCP flow groups. The Linux TCP implementation has a structure named `sock` to manage socket information for each connection and `tcp_opt` for TCP specific information. We added new variables to those structures for TCP-P: a pointer that points `flow_num` variable of its group is added to `tcp_opt` to get the number of flows of its group without searching the group list, and a pointer to the next `sock` structure in the same group is added in the `sock` structure. Each *group* structure has a pointer to its first member's structure `sock` and

[2] In this implementation, we use destination address as group classifier. Other classifications could be used just as well.

has a integer variable `flow_num` to count the number of member flows of the group. Whenever a new member TCP is added or deleted in a group, this count variable updates the number of member TCP flows. For a system-wide control of *strength* we added a new system control parameter `sysctl_tcp_strength` in `net/ipv4/sysctl_net_ipv4.c`. This parameter can be easily changed in the run-time using the `sysctl` system call.

3.2 Implementing Increase

In slow-start mode, the congestion window of each TCP-P flow in a group is increased by $\frac{k}{N}$ per non-duplicate ACK, as described in Equation (2). The amount of increase, $\frac{k}{N}$, is less than or equal to 1 when k is not bigger than N. Since floating point arithmetic is not supported in the Linux kernel, we let each TCP-P flow in our scheme increase its congestion window size by k after receiving N non-duplicate ACKs.

In congestion avoidance mode, each TCP-P flow in a group of size N with strength parameter k should increase its congestion window by 1 after receiving $\frac{\sum_{i=0}^{N-1}(\sum_{j=1}^{N} W_j + i)}{k*N}$ non-duplicate ACKs as Equation (4). To implement this for each TCP-P flow independently, we use the following equation:

$$\begin{aligned} &\frac{\sum_{i=0}^{N-1}\sum_{j=1}^{N} W_j + \sum_{i=0}^{N-1} i}{k*N} \\ &= \frac{N\sum_{j=1}^{N} W_j + \sum_{i=0}^{N-1} i}{k*N}. \end{aligned} \tag{7}$$

Therefore, each TCP-P flow should increase its congestion window size W_j by 1 after $\frac{N\sum_{j=1}^{N} W_j + \sum_{i=0}^{N-1} i}{k*N}$ non-duplicate ACKs. Alternatively, it can increase the congestion window by $\frac{k*N}{N\sum_{j=1}^{N} W_j + \sum_{i=0}^{N-1} i}$ per non-duplicate ACK.

Looking up other TCP-P flows' congestion window size at every non-duplicate ACK arrival may result in serious overhead. To reduce operation cost we assume that all TCP-P flows in a group have the same window size W_0, so that $\sum_{j=1}^{N} W_j = NW_0$. With this assumption, each flow does not need to know other TCP-P flows' congestion window sizes. Instead, it can use its own congestion window W_j to estimate the total window size for the group. Each TCP-P can find the size of its group, N, easily because each TCP-P structure has a pointer to its group's member count variable `flow_num` as shown in Fig. 1.

Since floating point arithmetic is not supported in Linux kernel, we increase the congestion window size of each flow by k after it received $\frac{N^2*W_j+\sum_{i=0}^{N-1} i}{N}$ non-duplicate ACKs. This number can be further simplified as follows:

$$\begin{aligned} &\frac{N^2 * W_j + \sum_{i=0}^{N-1} i}{N} \\ &= NW_j + \frac{(N-1)N}{2N} \\ &= NW_j + \frac{N-1}{2}. \end{aligned} \tag{8}$$

Therefore, each TCP-P flow in a group of size N and strength k should increase its congestion window by k after receiving $NW_j + \frac{N-1}{2}$ non-duplicate ACKs.

3.3 Implementing Decrease

TCP-P controls the decrease amount of total congestion window sizes of parallel TCP-P flows according to Equation (6) to match with that of k unmodified parallel TCP flows. One possible method to implement this is to decrease every TCP flow's congestion window by the same proportion. However, in this paper, when a TCP-P flow i detects a congestion event at time t and the elapsed time is not less than its minimum *srtt*, it responds like a normal TCP: It enters recovery mode and halves its own congestion window regardless of other parallel flows. Therefore, other TCP-P flows in the group can reduce their congestion window sizes less than the proportion shown in Equation (6) when the strength is $k > 1$.

Hence, the amount of decreases of congestion window sizes of other member TCP-P flows' become as follows:

$$W_j(t+) = W_j(t) * (\frac{1}{2} + \frac{N(k-1)}{2(N-1)k}),\ \forall j \neq i. \tag{9}$$

This equation is derived from the following equation to distribute the remaining amount of congestion window decrease among the other member TCP-P flows:

$$\begin{aligned}
&\sum_{j=1}^{N} W_j(\frac{2k-1}{2k}) \\
&= W_0 N(\frac{1}{2} + \frac{k-1}{2k}) \\
&= W_0(1 + N - 1)(\frac{1}{2} + \frac{k-1}{2k}) \\
&= \frac{1}{2}W_0 + W_0(N-1)(\frac{1}{2} + \frac{N(k-1)}{2(N-1)k}).
\end{aligned} \tag{10}$$

4 Evaluation

For the evaluation of TCP-P, we used the topology shown in Fig. 2. To emulate delays and packet losses in the Internet, we use NIST Net Emulator [12]. The NIST Net Emulator is implemented on a Linux machine and emulates the Internet by appropriately delaying and dropping packets. Because the network links in our experiments are fairly high-bandwidth (default 100Mbps) and the NIST Net delay parameters are large (50msec round-trip propagation delay,) we set the TCP parameters of the Linux end systems - such as `tcp_wmem` and `tcp_rmem` sizes - appropriately, rather than using system defaults. We also disabled the TCP time-stamping and TCP-Sack options to see the effects of our modification more clearly. By disabling TCP-Sack, Linux TCP works based on TCP-NewReno.

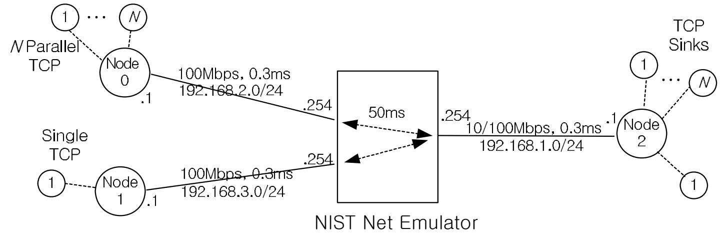


Fig. 2. Experiment Network

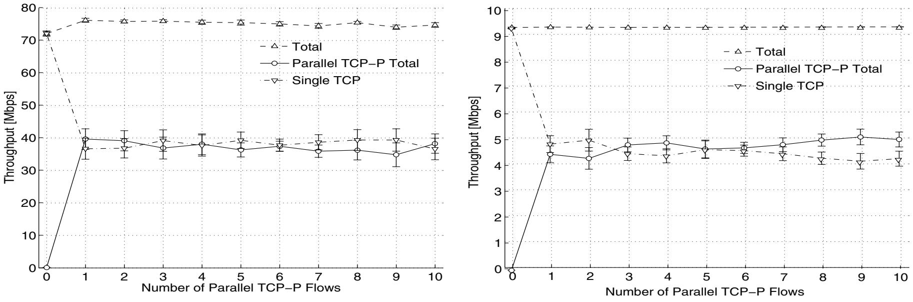
All the end-host nodes and the NIST Net Emulator are running on Linux PCs. The PCs we use for experiments are Pentium 4 or 3 machines with 10/100 Mbps Fast Ethernet network interface cards. Each Fast Ethernet card has an output queue of length 100 packets by default, and can be controlled if needed. Two TCP sender nodes, Node 0 and Node 1, run both on Redhat 9.0 with kernel 2.4.20-8. In machine Node 0 we installed a modified Linux kernel that supports. NIST Net Emulator and the TCP sink, Node 2, are running on Redhat Linux 7.2 with kernel 2.4.7-10.

For traffic generation and throughput measurements we use `iperf` [13], which supports parallel TCP flows and offers great flexibility for measurements. In all experiments in this section, every experiment was done for 100 sec to get an average value and repeated 10 times with 5 sec waiting time after each experiment unless told otherwise. Error bars in figures of this section represent 95% Confidence Interval of the data.

4.1 TCP Flow Groups with Strength $k = 1$

We first show the performance of TCP-P with $k = 1$. We open a group of parallel TCP-P flows from modified Linux kernel at Node 0 to a TCP sink Node 2 for 100 seconds, and a single unmodified TCP flow from another sender Node 1 to Node 2 for the same time. Fig. 3 (a) shows the experimental results with a varying number of parallel TCP-P flows from Node 0 with $k = 1$. This figure shows that the average of aggregated throughput of a group of parallel TCP-P flows of $k = 1$ with group size from 1 to 10 remains comparable to the average throughput of the single TCP flow from Node 1. In order to investigate the robustness of the TCP-P approach we repeated the experiments with a reduced bottleneck link speed by limiting the link speed from NIST Net Emulator to TCP sink node Node 2 from 100Mbps to 10Mbps. Fig. 3 (b) shows the experiment results with 10Mbps link. These results also show that TCP-P can regulate the aggressiveness of parallel TCP-P flows not to steal bandwidth from the single TCP flow, so that the throughput of the single TCP flow from Node 1 is comparable to that of total parallel TCP-P flows from Node 0 regardless of the group size N.

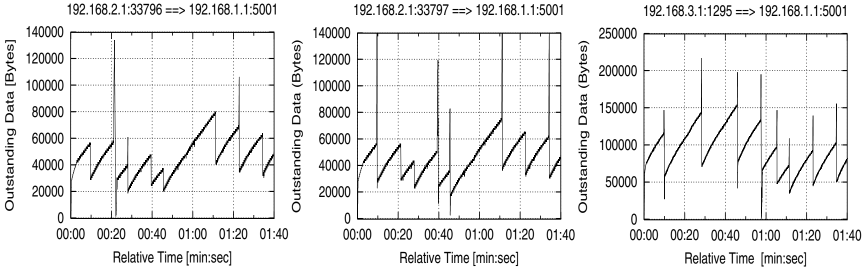
Figures in Fig. 4 illustrates some of the details of the operation of TCP-P. These figures are generated by `tcptrace` using `tcpdump` data on the sender Node 0 and Node 1. For the simplicity of comparison of the time-dependent behaviors



(a) 100Mbps Link

(b) 10Mbps Link

Fig. 3. Effect of N TCP-P Flows with $k = 1$ on a Single TCP Flow



(a) Flow 0

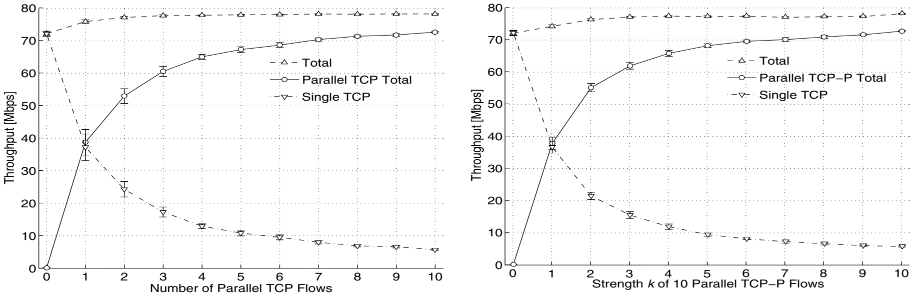
(b) Flow 1

(c) Flow 2

Fig. 4. Observed Outstanding Packets of Flows

we open two parallel TCP-P flows with $k = 1$, Flow 0 and Flow 1, from Node 0 to Node 2, and one TCP flow, Flow 2, from Node 1 to Node 2. Other conditions are the same to the previous experiment, and all connections start at the same time and finish after 100 seconds. Average throughput achieved by the two TCP-P flows from Node 0 and the single TCP flow from Node 1 were 4.73Mbps and 4.78Mbps, respectively.

The figures show the amount of outstanding data of each TCP flow, from which we can infer the changes of congestion windows of TCP flows. The spikes in the figures represent Fast-Retransmission behaviors of TCP flows. Fig. 4 (a) and Fig. 4 (b) are for two TCP-P flows from Node 0. We can see in these figures that there are congestion window decreases *without* spikes, which indicates adjustments of the congestion window by the other TCP-P in the group. Compared to these two figures, the change in the congestion window of Flow 2 in Fig. 4 (c) always have a spike before a reduction.



(a) Unmodified k Parallel TCP

(b) 10 Parallel TCP-P for different k

Fig. 5. Effects of Unmodified k TCP Flows and 10 TCP-P Flows with varying Strength k

4.2 TCP Flow Groups with Strength $k > 1$

In the following, we illustrate how TCP-P effectively controls the magnitude of aggressiveness of parallel TCP flows according to the strength parameter k. We first present experiment results with unmodified parallel TCP flows in Fig. 5 (a). Node 0 opens k unmodified parallel TCP flows to Node 2 for 100 seconds, while Node 1 opens a single TCP flow to Node 2 for the same time. Fig. 5 (a) shows the average throughput of TCP flows from Node 0 and Node 1 for a varying number of unmodified TCP flows from Node 0. The single TCP flow from Node 1 achieves increasingly smaller throughput with increasing numbers of unmodified parallel TCP flows from Node 0. It illustrates the unfairness of parallel TCP flows mentioned in Sec. 1.

In comparison, Fig. 5 (b) shows the results of TCP-P in the same environment, except that we let Node 0 open a group of 10 parallel TCP-P flows to Node 2 with varying strength k. Fig. 5 (b) shows average throughput of parallel TCP-P flows and a single TCP flow when we control the aggressiveness of the group of parallel TCP-P flows. In the figure, with $k = 0$ we describe the case of no parallel TCP-P flows sending any traffic to the destination, so that only the single flow from Node 1 consumes all bandwidth. By comparing (a) and (b) in Fig. 5 we see that TCP-P scheme accurately controls the overall aggressiveness of a group of 10 parallel TCP-P flows according to k. 10 TCP-P flows with strength k show almost the same effect to a single TCP as k unmodified parallel TCP flows.

The steady-state throughput models for N parallel TCP and TCP-P flows with strength k have been derived and interested readers can refer to them in [14].

5 Conclusion

In this paper, we proposed TCP-P for aggregate control of parallel TCP flows. TCP-P scheme uses *strength* as a single - easily tunable - parameter to accurately control the aggressiveness of a group of TCP flows with respect to a single flow sharing the same bottleneck link. We showed that by employing TCP-P we can control the total aggressiveness or unfairness of parallel TCP flows against TCP flows from other nodes in a easily parameterizable and controllable way without requiring application modification. For future work, we are considering an adaptive control of the strength of parallel TCP flows.

References

1. Tom Dunigan: Net 100 Project (2004) URL: `http://www.csm.ornl.gov/~dunigan/netperf/parallel.html`.
2. S. Floyd and K. Fall: Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking **7** (1999) 458–472
3. D. Maltz and P. Bhagwat: TCP Splicing for Application Layer Proxy Performance. IBM Research Report RC 21139 (1998)
4. H. Balakrishnan, H. Rahul, and S. Seshan: An Integrated Congestion Management Architecture for Internet Hosts. In: ACM SIGCOMM. (1999)
5. Thomas Hacker, Brian Noble, and Brian Athey: Improving Throughput and Maintaining Fairness using Parallel TCP. In: Infocom. (2004)
6. Y. Gao, G. He, C. Hou, and S. Paul: COCOON: an alternate approach to end-host congestion management. submitted to IEEE Trans. on Computers (2002) URL: `stat.bell-labs.com/who/yuangao/papers/cocoon.pdf`.
7. J. Crowcroft and P. Oechslin: Differentiated end-to-end Internet Services using a Weighted Proportionally Fair Sharing TCP. ACM CCR **28** (1998) 53–69
8. V. Paxson: End-to-End Internet packet dynamics. In: ACM SIGCOMM. (1997)
9. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow: TCP Selective Acknowledgement Options. RFC-2018 (1996)
10. P. Sarolahti and A. Kuznetsov: Congestion Control in Linux TCP. In: Proceedings of Usenix 2002/Freenix Track. (2002)
11. Matt Mathis, Jeff Semke, Jamshid Mahdavi, and Kevin Lahey: The Rate Halving Algorithm for TCP Congestion Control (1999) URL: `http://www.psc.edu/networking/rate_halving.html`.
12. M. Carson and D. Santay: Tools: NIST Net: a Linux-based network emulation tool. ACM CCR **33** (2003) 111–126
13. Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson, and Kevin Gibbs: Iperf Version 1.7.0 (2003) URL:`http://www.noc.ucf.edu/Tools/Iperf/`.
14. Soohyun Cho and Riccardo Bettati: Aggregate Control of Parallel TCP flows. Technical Report TAMU-CS-2004-11-1 (2004)

Multi-level Dynamic Guard Channels for Priority Access in Cellular Systems

Tung Chong Wong[1], Jon W. Mark[2], and Kee Chaing Chua[3]

[1] Institute for Infocomm Research, 21 Heng Mui Keng Terrace,
Singapore 119613, Singapore
wongtc@i2r.a-star.edu.sg
[2] Center for Wireless Communications, University of Waterloo,
Waterloo, Ontario, Canada N2L 3G1
jwmark@bbcr.uwaterloo.ca
[3] Electrical and Computer Engineering, National University of Singapore,
10 Kent Ridge Crescent, Singapore 119260, Singapore
chuakc@nus.edu.sg

Abstract. A multi-level dynamic guard channels (MLDGC) scheme to efficiently provide priority access for handoff calls over new calls in cellular systems is proposed. The switch between levels is controlled by the number of dropped calls. Each additional level has an increasing number of guard channels as triggered by the number of dropped calls. An analytical formulation of the steady state probabilities, new call blocking probability, handoff call blocking probability and system utilization is presented. The handoff call arrival rate is iteratively obtained. Numerical results show that the performance of MLDGC is almost the same as a fixed guard channels (FGC) scheme under light load. However, the advantage of the MLDGC scheme is demonstrated under heavy load where the handoff call blocking probability is much better than that of a FGC scheme, giving more priority and protection to handoff calls over new calls.

1 Introduction

In mobile communications, efficient handoff to minimize the number of handoff call dropping is critically important. One of the most fundamental approaches to provide access for handoff calls over new calls is the fixed guard channels (FGC) scheme [1]. By reserving a small number of guard channels for the exclusive use by handoff calls, the handoff call blocking probability is significantly reduced at the expense of a slight increase in the new call blocking probability. Thus priority is given to handoff calls over new calls as new call blockings are more acceptable to users than handoff call blocking. That is, once a call is admitted, a user is more likely to object to being dropped in the middle of a call than when a user cannot get into the system at the start of a new call.

Dynamic guard channels (DGC) schemes have been proposed and studied [2-6]. Refs. [2-3] consider a DGC scheme which adapts the number of guard channels in each radio cell according to the current estimate of the handoff call arrival rate. This

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 598 – 609, 2005.

rate is derived based on the current number of ongoing calls in the neighbouring cells and the mobility pattern. A DGC scheme which adapts to the traffic load is proposed in [4]. Another DGC scheme which varies the number of guard channels based on the current number of users in a reservation cluster is proposed in [5]. Ref. [6] models a FGC scheme and a DGC scheme for mobile transactions. The performances of the DGC schemes in [2-4] are evaluated using *computer simulations*, while those in [5-6] are evaluated using *numerical techniques*.

In this paper, we propose a multi-level DGC (MLDGC) scheme. This MLDGC scheme has multiple levels of states that adapt to varying number of guard channels. Each additional level has an increasing number of guard channels as triggered by the *number of dropped handoff calls*. The greater the number of dropped handoff calls, the higher the level in the MLDGC scheme. The higher the level, the greater the number of guard channels. The advantage of MLDGC is that the handoff call blocking probability is much better than that of a FGC scheme under heavy traffic load, giving higher access priority and protection to handoff calls over new calls. An *analytical formulation* of the steady state probabilities, new call blocking probability, handoff call blocking probability and system utilization is presented in the sequel. The handoff call arrival rate is iteratively obtained.

The rest of the paper is organized as follows. Section 2 describes the system model, system level and call level parameters. Section 3 describes the proposed MLDGC scheme. Section 4 presents an analytical model for the MLDGC scheme. Numerical results for a two-level MLDGC scheme are presented in Section 5. Finally, concluding remarks are made in Section 6.

2 System Model

Without loss of generality, we consider a cellular system consisting of square cells. The following system parameters are used throughout the paper.

System Level Parameters

C: total number of channels in a cell

C_{Gk}: number of guard channels in level k of the MLDGC scheme in a cell, k=1,2,…,K, and $C_{G1}< C_{G2}<\ldots< C_{GK}$

K: number of levels in the MLDGC scheme

The call level parameters used throughout the paper are listed below.

Call Level Parameters

B_n : new call blocking probability

B_h : handoff call blocking probability

N_u: system utilization

λ_n : arrival rate of new calls

λ_h : arrival rate of handoff calls

$\lambda = \lambda_n + \lambda_h$: mean call arrival rate of new and handoff calls

μ_c^{-1} : mean call holding time of a call in a cell

μ_h^{-1} : mean dwell time (interhandoff time) of a call in a cell

$\mu = \mu_c + \mu_h$: mean equivalent service rate of a call in a cell

The dynamics of a radio cell is driven by new call requests, call terminations, and handoffs induced by user mobility. Maintaining an ongoing call is more important than admitting a new call. Hence, handoff calls should be given a higher access priority, or a lower blocking probability than new calls.

3 Multi-level Dynamic Guard Channel (MLDGC) Scheme

The MLDGC scheme works as follows. There are *K* levels in the MLDGC scheme. In level 1, the MLDGC scheme works similarly to a FGC scheme. A small number of guard channels, C_{G1}, in a cell is reserved for exclusive use by handoff calls, while the other channels in the cell can be used by both the handoff calls and the new calls. However, the MLDGC scheme differs from the FGC scheme in that the former will be triggered to move to the next level (level 2) when a handoff call is dropped because there is no available channel in the cell. In level 2, more guard channels, C_{G2} ($>C_{G1}$), are allocated for exclusive use by handoff calls than in level 1. When these guard channels are no longer occupied by handoff calls, the scheme will switch back to level 1. Similarly, when another handoff call is dropped due to full channel occupancy while in level 2, the MLDGC scheme will move to the next level (level 3) with even more guard channels, C_{G3} ($>C_{G2}> C_{G1}$), allocated in this new level. Similarly, if these guard channels are no longer occupied by handoff calls, the scheme will switch back to level 1. This dynamic allocation of guard channels can continue up to a maximum of *K* levels. At the *K*th level, the number of guard channels is, C_{GK}, such that $C_{GK} >\ldots>C_{G3} >C_{G2}> C_{G1}$.

Fig. 1 shows a two-dimensional finite state Markov chain for the MLDGC scheme. State (*k*,*i*) represents the *k*th level of the MLDGC scheme and there are *i* channels in service in the cell. $\lambda\,(=\lambda_n+\lambda_h)$ is the mean call arrival rate of new and handoff calls in a cell, while λ_n and λ_h are respectively the arrival rate of new calls and handoff calls in the cell. $\mu\,(=\mu_c+\mu_h)$ is the mean equivalent service rate of a call in a cell, while μ_c^{-1} and μ_h^{-1} are respectively the mean call holding time of a call and the mean dwell time (interhandoff time) of a call in the cell. In level 1 of the MLDGC scheme, only handoff calls can be admitted in the C_{G1} channels as indicated by the transition rates of λ_h in Fig. 1. On the other hand, the rest of the channels can be used by both the new and handoff calls as indicated by the transition rates of λ. If the MLDGC scheme is in state (*k*,*C*), *k*=1,2,…*K*-1, it will be triggered to jump to the state (*k*+1,*C*) in the next level of (*k*+1) by a handoff call arriving to a fully occupied cell. This transition is indicated by the state transition rate of λ_h from state (*k*,*C*) to state (*k*+1,*C*). For every jump to the next level, the number of guard channels is increased such that $C_{GK} >\ldots>C_{G3} >C_{G2}> C_{G1}$. Note that in levels 2 to *K*, only handoff calls can use the respective level's guard channels as indicated by the transition rates of λ_h. In state (k,i), the channels are serviced at a mean rate of $i\mu$ and it will jump to state (k,i-1) due to a call termination or a call handoff to another cell. If the MLDGC scheme is in state (k,C-C_{Gk}) in level k, k=2,3,…,K, it will return to state (1,C-C_{Gk}-1) in level 1 with a transition rate of (C-C_{Gk})μ if a channel has been served due to call termination or call handoff to another cell before any handoff call arrival.

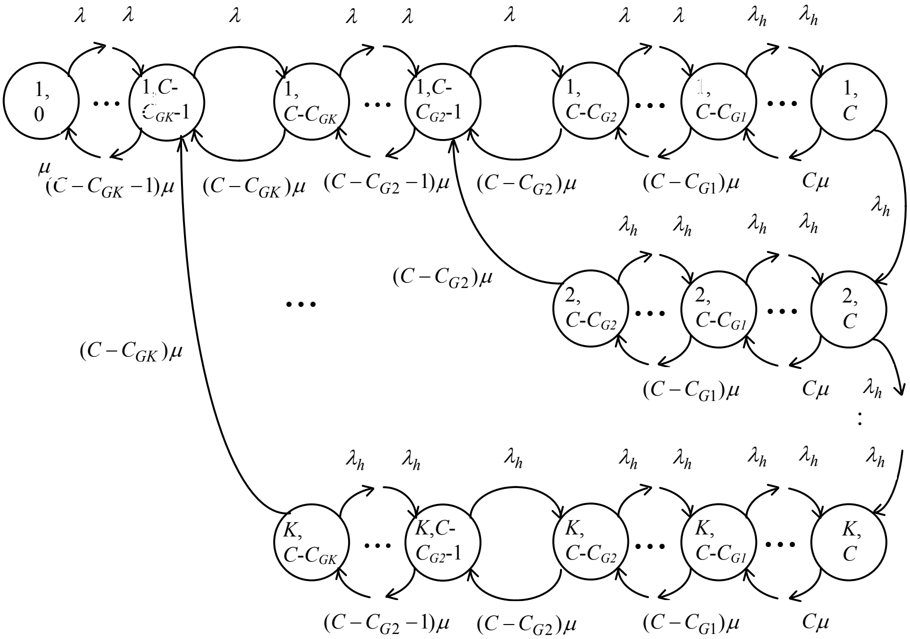


Fig. 1. State transition diagram for the MLDGC scheme

4 Analytical Model

Consider a 2-level MLDGC model, that is, K=2. The guard channels, C_{Gk}, are reserved for handoff connections only. To facilitate analytical modeling, it is necessary to make certain assumptions about the traffic parameters. It is not unreasonable to assume that the holding time has a negative exponential distribution [7]. Although a negative exponential distribution assumption may not be as reasonable for the cell dwell time, for analytical tractability, we will make the same assumption for cell dwell time (interhandoff time) [7] and model the channel occupancy as a 2-dimensional Markov chain with the call level transition rate parameters as in Fig. 1. This Markov chain can be modeled and solved using numerical techniques. Instead of solving this Markov chain using numerical techniques, this Markov chain is solved *analytically* by equating probability flows in this section.

Let $P_{k,i}$ be the steady state probabilities in state (k,i) of the Markov chain, where k=1,…,K, and i=0,…,C for k=1 and i=C-C_{Gk},…,C for k=2,…,K. Solving the Markov chain in Fig. 1 by equating probability flows across surfaces of the Markov chain and using the sum of total state probabilities as one, we get

$$P_{1,i}=P_{1,0}\left(\frac{\lambda}{\mu}\right)^{i}\frac{1}{i!},\qquad 1\le i\le C-C_{G2}-1,\quad (1)$$

$$\begin{aligned}P_{1,i}={}&P_{1,0}\,\frac{1}{X}\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1}\frac{1}{(C-C_{G2}-1)!}\bigg[\frac{\lambda_h}{\lambda}+\frac{(C-j+1)\mu}{\lambda}\bigg[\frac{\lambda_h}{\lambda}+\frac{(C-j+2)\mu}{\lambda}\bigg[\ldots\\&\times\bigg[\frac{\lambda_h}{\lambda}+\frac{(C-C_{G1})\mu}{\lambda}\bigg[1+\frac{(C-C_{G1}+1)\mu}{\lambda_h}\bigg[1+\frac{(C-C_{G1}+2)\mu}{\lambda_h}\bigg[\ldots\ \bigg[1+\frac{(C-1)\mu}{\lambda_h}\\&\times\bigg[1+\frac{C\mu}{\lambda_h}\bigg]\ \bigg]\ldots\ \bigg]\bigg]\bigg]\bigg]\ldots\ \bigg]\bigg]\bigg],\qquad C-C_{G2}-1\le i\le C-C_{G1}-1,\ j=C-i,\end{aligned}\quad (2)$$

$$\begin{aligned}P_{1,i}={}&P_{1,0}\,\frac{1}{X}\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1}\frac{1}{(C-C_{G2}-1)!}\bigg[1+\frac{(C-j+1)\mu}{\lambda_h}\bigg[1+\frac{(C-j+2)\mu}{\lambda_h}\bigg[\ldots\\&\times\bigg[1+\frac{(C-1)\mu}{\lambda_h}\bigg[1+\frac{C\mu}{\lambda_h}\bigg]\ \bigg]\ldots\ \bigg]\bigg]\bigg],\qquad C-C_{G1}\le i\le C-1,\ j=C-i,\end{aligned}\quad (3)$$

$$P_{1,i}=P_{1,0}\,\frac{1}{X}\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1}\frac{1}{(C-C_{G2}-1)!},\qquad i=C,\quad (4)$$

$$P_{2,i}=P_{1,0}\,\frac{1}{X}\left(\frac{\lambda_h}{(C-C_{G2})\mu}\right)\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1}\frac{1}{(C-C_{G2}-1)!},\qquad i=C-C_{G2},\quad (5)$$

$$\begin{aligned}P_{2,i}={}&P_{1,0}\,\frac{1}{X}\left(\frac{\lambda_h}{(C-C_{G2})\mu}\right)\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1}\frac{1}{(C-C_{G2}-1)!}\bigg[\frac{C-C_{G2}}{C-j}\bigg[1+\frac{\lambda_h}{(C-j-1)\mu}\\&\times\bigg[1+\frac{\lambda_h}{(C-j-2)\mu}\bigg[\ldots\ \bigg[1+\frac{\lambda_h}{(C-C_{G2}+2)\mu}\bigg[1+\frac{\lambda_h}{(C-C_{G2}+1)\mu}\\&\times\bigg[1+\frac{\lambda_h}{(C-C_{G2})\mu}\bigg]\bigg]\bigg]\ldots\ \bigg]\bigg]\bigg]\bigg],\qquad C-C_{G2}+1\le i\le C,\ j=C-i.\end{aligned}\quad (6)$$

where

$$\begin{aligned}X={}&\bigg[\frac{\lambda_h}{\lambda}+\frac{(C-C_{G2})\mu}{\lambda}\bigg[\frac{\lambda_h}{\lambda}+\frac{(C-C_{G2}+1)\mu}{\lambda}\bigg[\ldots\ \bigg[\frac{\lambda_h}{\lambda}+\frac{(C-C_{G1})\mu}{\lambda}\\&\times\bigg[1+\frac{(C-C_{G1}+1)\mu}{\lambda_h}\bigg[1+\frac{(C-C_{G1}+2)\mu}{\lambda_h}\bigg[\ldots\ \bigg[1+\frac{(C-1)\mu}{\lambda_h}\\&\times\bigg[1+\frac{C\mu}{\lambda_h}\bigg]\ \bigg]\ldots\ \bigg]\bigg]\bigg]\bigg]\ldots\ \bigg]\bigg]\bigg],\end{aligned}\quad (7)$$

and

$$
\begin{aligned}
P_{1,0} = \Bigg\{ & 1 + \sum_{i=1}^{C-C_{G2}-1} \left(\frac{\lambda}{\mu}\right)^{i} \frac{1}{i!} + \sum_{j=C_{G1}+1}^{C_{G2}} \frac{1}{X}\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1} \frac{1}{(C-C_{G2}-1)!} \\
& \times \Big[\frac{\lambda_h}{\lambda} + \frac{(C-j+1)\mu}{\lambda}\Big[\frac{\lambda_h}{\lambda} + \frac{(C-j+2)\mu}{\lambda}\Big[\dots\Big[\frac{\lambda_h}{\lambda} + \frac{(C-C_{G1})\mu}{\lambda}\Big[1 + \frac{(C-C_{G1}+1)\mu}{\lambda_h} \\
& \times \Big[1 + \frac{(C-C_{G1}+2)\mu}{\lambda_h}\Big[\dots\Big[1 + \frac{(C-1)\mu}{\lambda_h}\Big[1 + \frac{C\mu}{\lambda_h}\Big]\Big]\dots\Big]\Big]\Big]\Big]\dots\Big]\Big]\Big] \\
& + \sum_{j=1}^{C_{G1}} \frac{1}{X}\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1} \frac{1}{(C-C_{G2}-1)!}\Big[1 + \frac{(C-j+1)\mu}{\lambda_h}\Big[1 + \frac{(C-j+2)\mu}{\lambda_h}\Big[\dots \\
& \times \Big[1 + \frac{(C-1)\mu}{\lambda_h}\Big[1 + \frac{C\mu}{\lambda_h}\Big]\Big]\dots\Big]\Big]\Big] + \frac{1}{X}\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1} \frac{1}{(C-C_{G2}-1)!} \\
& + \frac{1}{X}\left(\frac{\lambda_h}{(C-C_{G2})\mu}\right)\left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1} \frac{1}{(C-C_{G2}-1)!} + \sum_{j=0}^{C_{G2}-1} \frac{1}{X}\left(\frac{\lambda_h}{(C-C_{G2})\mu}\right) \\
& \times \left(\frac{\lambda}{\mu}\right)^{C-C_{G2}-1} \frac{1}{(C-C_{G2}-1)!}\Big[\frac{C-C_{G2}}{C-j}\Big[1 + \frac{\lambda_h}{(C-j-1)\mu}\Big[1 + \frac{\lambda_h}{(C-j-2)\mu}\Big[\dots \\
& \times \Big[1 + \frac{\lambda_h}{(C-C_{G2}+2)\mu}\Big[1 + \frac{\lambda_h}{(C-C_{G2}+1)\mu}\Big[1 + \frac{\lambda_h}{(C-C_{G2})\mu}\Big]\Big]\Big]\dots\Big]\Big]\Big]\Big]\Bigg\}^{-1}
\end{aligned}
\tag{8}
$$

The new call blocking probability, B_n, is given by

$$B_n = \sum_{i=C-C_{G1}}^{C} P_{1,i} + \sum_{i=C-C_{G2}}^{C} P_{2,i}. \tag{9}$$

The handoff call blocking probability, B_h, is given by

$$B_h = P_{1,C} + P_{2,C}. \tag{10}$$

The system utilization, N_u, is given by

$$N_u = \sum_{i=1}^{C} iP_{1,i} + \sum_{i=C-C_{G2}}^{C} iP_{2,i} = \frac{\lambda_n(1-B_n)+\lambda_h(1-B_h)}{\mu}. \tag{11}$$

Equating the handoff call arrival rate to the product of the average handoff rate for a call and the average number of calls, we have

$$\lambda_h = \mu_h N_u = \mu_h \times \frac{(1-B_n)\lambda_n + (1-B_h)\lambda_h}{\mu_c + \mu_h} = \frac{\mu_h(1-B_n)\lambda_n}{\mu_c + B_h\mu_h}. \tag{12}$$

Thus, the handoff call arrival rate can be approximated under low blocking probabilities as follows:

$$\lambda_h = \mu_h \lambda_n / \mu_c \ , \tag{13}$$

where $\mu_h = v / s$, v is the speed of the class k mobile and s is the cell length of a square cell.

Note that the steady state probabilities, $P_{k,i}$'s, are functions of the handoff call arrival rate, λ_h, and λ_h is a function of the system utilization, N_u, which is a function of $P_{k,i}$'s. We use an iterative method to determine the λ_h and the $P_{k,i}$'s as follows.

1. Initialize the handoff call arrival rate, $\lambda_h^{(0)} = \mu_h \lambda_n / \mu_c$, where the superscript in parentheses means step 0.
2. Iterate between $P_{k,i}$'s as in (1)-(8) for all k and i, N_u as in (11) and λ_h as in (12) until

$$\lambda_h^{(n)} - \lambda_h^{(n-1)} < \varepsilon_{\lambda_h} , \tag{14}$$

where the superscript n in the parentheses denotes the nth iteration step, and ε_{λ_h} is the error thresholds for λ_h. Using this iterative method, the handoff call arrival rate, λ_h, is obtained iteratively. Note that the initial handoff call arrival rate, $\lambda_h^{(0)}$, is set to the approximate handoff call arrival rate under low blocking probabilities as in (13).

5 Numerical Results

In this section we present results to examine the performance of new and handoff call blocking probabilities for the MLDGC scheme as well as the performance in system utilization. We consider an MLDGC scheme with 2 levels (K=2). The numerical results for the MLDGC scheme have been obtained by means of the foregoing analysis from Section 4, while those for the FGC scheme have been obtained from the analysis in the Appendix. The parameter values used in the numerical example presented in this section are tabulated in Table 1. The parameter values used are for illustration purposes only.

Table 1. Parameter Values Used

	C=10	C=20	C=30
$1/\mu_c$	1 minute	1 minute	1 minute
v	36 km/hr	36 km/hr	36 km/hr
s	200 m	200 m	200 m
$1/\mu_h$	1/3 minute	1/3 minute	1/3 minute
C_{G1}	2	4	6
C_{G2}	4, 6, 8	8, 12, 16	12, 18, 24
ε_{λ_h}	1×10^{-10}	1×10^{-10}	1×10^{-10}

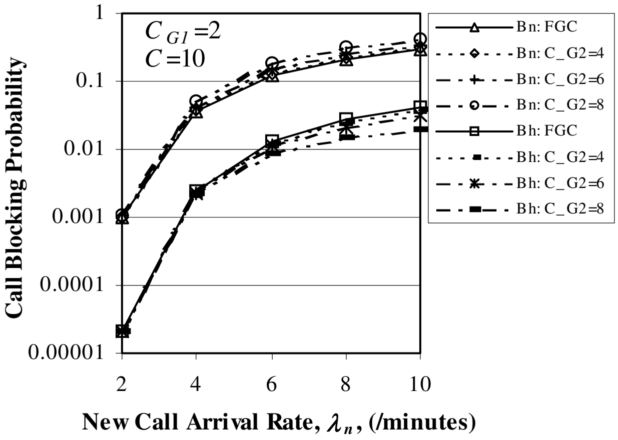


Fig. 2. Call blocking probabilities with C=10

The new and handoff call blocking probabilities for the FGC scheme and the proposed MLDGC scheme as a function of the new call arrival rate with C=10, C=20 and C=30 are shown in Figs. 2, 3 and 4, respectively. For illustration purpose, guard channels C_{G1}=2, C_{G1}=4, and C_{G1}=6 are chosen for the FGC scheme and level 1 of the MLDGC scheme with C=10, C=20, and C=30, respectively. For the same purpose, C_{G2}={4,6,8}, C_{G2}={8,12,16} and C_{G2}={12,18,24} are varied for the MLDGC scheme with C=10, C=20, and C=30, respectively. It is observed that the performances of the new and handoff blocking probabilities in the MLDGC scheme are almost the same as those in the FGC scheme under light load conditions. However, the advantage of the

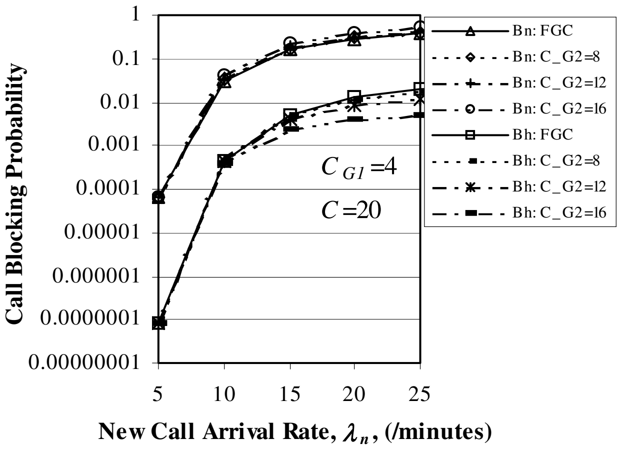


Fig. 3. Call blocking probabilities with C=20

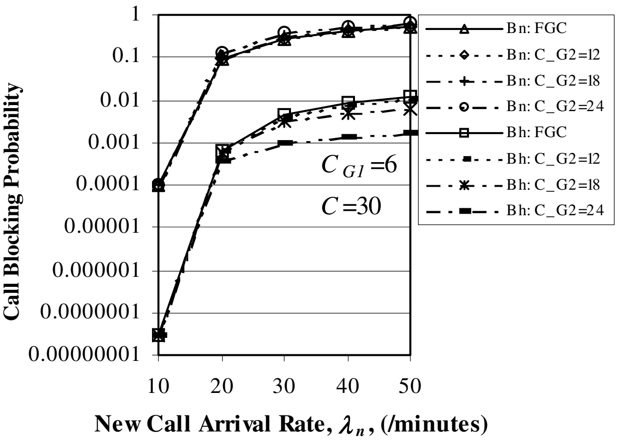


Fig. 4. Call blocking probabilities with C=30

MLDGC scheme is shown under the heavy load conditions where the performances of the handoff call blocking probabilities of the MLDGC scheme are decreasing lower than those of the FGC scheme with increasing number of guard channels in level 2, C_{G2}, of the MLDGC scheme. These gains in handoff call blocking are obtained at the expense of slight increases of the new call blocking probabilities. Thus the objective of giving higher priority to handoff calls over new calls is further achieved by the MLDGC scheme under heavy load. Hence, extra protection is given to handoff calls during heavy loads.

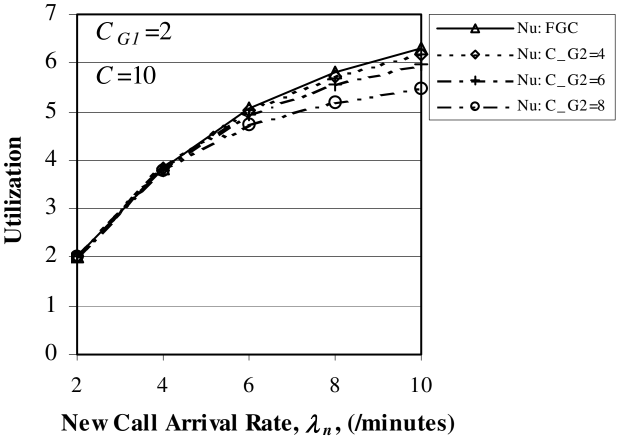


Fig. 5. System utilization with C=10

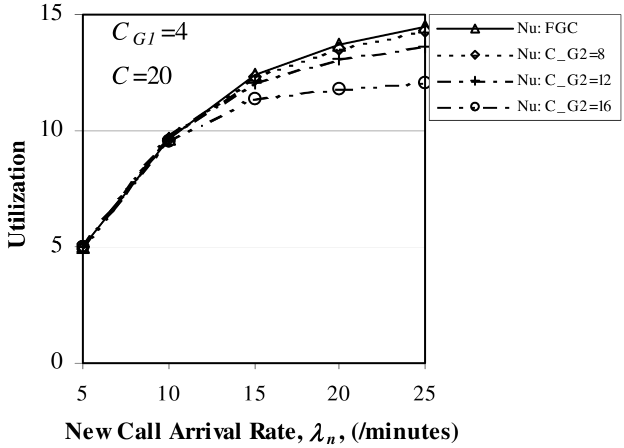


Fig. 6. System utilization with *C*=20

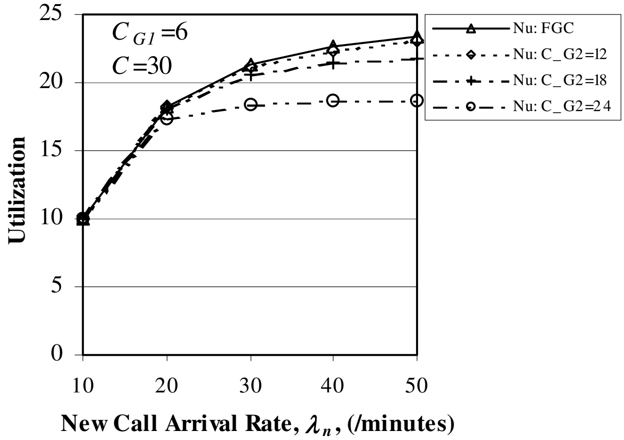


Fig. 7. System utilization with *C*=30

Figs. 5, 6 and 7 show the system utilizations for the FGC scheme and the proposed MLDGC scheme as a function of the new call arrival rate for *C*=10, *C*=20 and *C*=30, respectively. From these figure, it can be seen that the system utilizations for the MLDGC scheme are almost the same as those for the FGC scheme under light load. However, the system utilizations for the MLDGC scheme are lower than those of the FGC scheme under heavy loads. The reason for this is the increased number of guard channels in level 2. Thus this is the tradeoff for using the MLDGC scheme over the FGC scheme. However, as seen earlier in this section, the MLDGC scheme fulfils the objective of giving higher priority and protection to handoff calls over new calls under heavy load. Thus such a tradeoff is acceptable.

6 Concluding Remarks

A multi-level dynamic guard channels (MLDGC) scheme to enhance access priority for handoff calls in cellular systems is proposed in Section 3. Each additional level has an increasing number of guard channels as triggered by the number of dropped calls. An analytical formulation of the steady state probabilities, new call blocking probability, handoff call blocking probability and system utilization for a proposed multi-level dynamic guard channels scheme with 2 levels is presented in Section 4. The handoff call arrival rate is iteratively obtained. The performance of a 2-level MLDGC scheme with the number of channels in a cell at 10, 20 and 30 are illustrated in numerical examples in Section 5. These examples show that the performance of the MLDGC scheme behaves like a FGC scheme under light load conditions, while the performances of the handoff call blocking probabilities of the MLDGC scheme are decreasing lower than those of the FGC scheme with increasing number of guard channels in level 2 of the MLDGC scheme under heavy load conditions. Thus, the MLDGC scheme introduced in this paper shows promise in terms of fulfilling the objective of giving higher priority and protection to handoff calls over new calls under heavy load, at the expense of a slight degradation in system utilization.

References

1. Hong, D., Rappaport, S.S.: Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and Nonprioritized Handoff Procedures – Version 2b. CEAS Tech. Rep. 773, College of Engineering and Applied Sciences, State University of New York, Stony Brook, NY 11 794 USA (June 1, 1999)
2. Yu, O.T.W., Leung, V.C.M.: Self-Tuning Prioritized Call Handling Mechanism with Dynamic Guard Channels for Mobile Cellular Systems. IEEE VTC (1996) 1520-1524
3. Yu, O.T.W., Leung, V.C.M.: Adaptive Resource Allocation for Prioritized Call Admission over an ATM-Based Wirelss PCN. IEEE JSAC, Vol. 15, No. 7, (September 1997) 1208-1225
4. Wei, Y., Lin, C., Ren, F., Raad, R., Dutkiewicz, E.: Dynamic Priority Handoff Scheme in Differentiated QoS Wireless Multimedia Networks. IEEE ISCC (2003) 131-136
5. Lee, S.-H., Park, S.-W.: Handoff with Dynamic Channels in ATM-Based Mobile Networks. International Conference on Information Networking, (1998) 439-442
6. Chen, G.-C., Lee, S.-Y.: Modeling the Static and Dynamic Guard Channel Schemes for Mobile Transactions. International Conference on Parallel and Distributed Computing (1998) 258-265
7. Borst, S.C., Mitra, D.: Virtual partitioning for robust resource sharing: Computational techniques for heterogeneous traffic. IEEE JSAC, Vol. 16, (1998) 668-678

Appendix – Analysis for a Fixed Guard Channels Scheme

Let us consider only level 1 in Fig 1 and this is equivalent to a FGC scheme. Let P_i be the steady state probabilities in state i of the Markov chain, where i=0,1,…,C and k=1 is redundant. Solving this one-dimensional Markov chain by equating probability flows across surfaces of the Markov chain and using the sum of total state probabilities as one, we get

$$P_i = P_0 \left(\frac{\lambda}{\mu} \right)^i \frac{1}{i!}, \qquad 1 \le i \le C - C_{G1}, \tag{15}$$

$$P_i = P_0 \frac{\lambda^{C-C_{G1}} (\lambda_h)^{i-(C-C_{G1})}}{\mu^i i!}, \qquad C - C_{G1} + 1 \le i \le C, \tag{16}$$

where

$$P_0 = \left[\sum_{i=0}^{C-C_{G1}} \left(\frac{\lambda}{\mu} \right)^i \frac{1}{i!} + \sum_{i=C-C_{G1}+1}^{C} \frac{\lambda^{C-C_{G1}} (\lambda_h)^{i-(C-C_{G1})}}{\mu^i i!} \right]^{-1}. \tag{17}$$

The new call blocking probability, B_n, is given by

$$B_n = \sum_{i=C-C_{G1}}^{C} P_i . \tag{18}$$

The handoff call blocking probability, B_h, is given by

$$B_h = P_C . \tag{19}$$

The system utilization, N_u, is given by

$$N_u = \sum_{i=1}^{C} i P_i = \frac{\lambda_n (1 - B_n) + \lambda_h (1 - B_h)}{\mu}. \tag{20}$$

The handoff call arrival rate is iteratively obtained as in Section 4.

Slotted Aloha with Priorities and Random Power

E. Altman[1,⋆], D. Barman[2,⋆⋆] A. Benslimane, and R. El Azouzi[3]

[1] INRIA, BP93, 06902 Sophia-Antipolis, France
eitan.altman@sophia.inria.fr
[2] 111 Cummington Street, Dept. of Computer Science,
Boston University, Boston, MA 02215, USA
dhiman@cs.bu.edu
[3] LIA/CERI, Université d'Avignon, Agroparc, BP 1228, 84911, Avignon, France
{benslimane, rachid.elazouzi}@univ-avignon.fr

Abstract. We study distributed choice of retransmission probabilities in slotted Aloha under power differentiation. We consider random transmission powers and further study the role of priorities (through power control) given either to new arriving packets or to backlogged ones. We study both the cooperative team problem in which a common objective is jointly optimized as well as the noncooperative game problem.We show that the new proposed schemes not only improve the average performances considerably but are also able in some cases to eliminate the bi-stable nature of the slotted Aloha.

1 Introduction and Problem Formulation

Aloha [1] and slotted Aloha [11] have long been used as random distributed medium access protocols for radio channels. They are used in satellite networks and cellular telephone networks for the sporadic transfer of data packets. In these protocols, packets are transmitted sporadically by various users. If packets are sent simultaneously by more than one user then they collide. After a packet is transmitted, the transmitter receives the information on whether there has been a collision (and retransmission is needed) or whether it has been well received. All colliding packets are assumed to be corrupted which get backlogged and are retransmitted after some random time. We focus on the slotted Aloha [5] in which time is divided into units. At each time unit a packet may be transmitted, and at the end of the time interval, the sources get the feedback on whether there was zero, one or more transmissions (collision) during the time slot. A packet that arrives at a source is immediately transmitted. We introduce three new schemes in which multiple power levels are used for transmission. When several packets are sent simultaneously, one of them can often be successfully

⋆ Partially supported by the Euro-NGI network of excellence.
⋆⋆ Partially supported by INRIA through the ARC PRIXNET.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 610–622, 2005.

received due to the power capture effect. We assume that the packet with the largest received power captures the channel [7, 10, 12, 13]; if two or more packets are transmitted simultaneously with the same power, we assume that neither one can be captured. In addition to the power diversity, already proposed in [7, 10, 12, 13], we consider differentiation between new packets and backlogged packets allowing prioritization of one or the other in terms of transmitted power. We study and compare (1) a scheme with power diversity and without prioritization in transmission or retransmission; (2) a scheme in which a new packet is transmitted with the lowest power, and backlogged packets are transmitted at a random power selected among N larger distinct levels; (3) a scheme in which a new packet is transmitted with the highest power, and backlogged packets are transmitted at a random power selected among N lower distinct levels; and (4) standard slotted Aloha.

We first consider the optimal selection of transmission probabilities for the various schemes so as to maximize the throughput or minimize the expected delay. We discover that in heavy load, the optimality is obtained at the expense of huge expected delay of backlogged packets (EDBP). We therefore consider also the alternative objective of minimizing the EDBP. We also solve the multicriteria problem. We show that the new schemes not only improve the average performances considerably but also improve the stability performance.

In addition to the global optimization, we study the game problem in which each mobile chooses its transmission probability selfishly so as to optimize its individual objective. We show that the power diversity and the prioritization profit to mobiles also in this competitive scenario.

Various game formulations of slotted aloha with a single power have been derived and studied in [3, 4, 6, 8, 9] for the non-cooperative choice of transmission probabilities. Several papers study slotted aloha with power diversities but without differentiating between transmitted and backlogged packets, and without the game formulation [7, 10, 12, 13]. We consider a central receiver and m sources without buffer. We assume a perfect capture model where a successful capture of a packet at the receiver occurs when the power level (among N different levels) selected for this packet is greater than those of all other packets transmitted in the same slot.

We use a Markovian model extending [5–Sec. 4.2.2]. Packet arrivals to sources, independently of each other, follow Bernoulli process with parameter q_a. As long as there is a packet at a source (i.e. it has not been successfully transmitted) new packets to that source are blocked and lost.[1] A backlogged packet at source i is retransmitted with probability q_r^i that does not change with time. Since sources are symmetric, we shall further restrict to finding a symmetric optimal solution (i.e., $q_r^i = q,\ \forall i$).

We shall use as the state of the system the number of backlogged nodes (or equivalently, of backlogged packets) at the beginning of a slot, and denote it

[1] This assumption is equivalent to saying that a source does not generate new packets as long as a previous packet is not successfully transmitted.

frequently with n. For any choice of $q_r^i \in (0,1]$, the state process is a Markov chain that contains a single ergodic chain. Let $\mathbf{q}_r$ be the vector of retransmission probabilities for all users (whose jth entry is q_r^j). Let $\pi(\mathbf{q}_r)$ be the corresponding vector of steady state probabilities where its nth entry, $\pi_n(\mathbf{q}_r)$, denotes the probability of n backlogged nodes. When all entries of $\mathbf{q}_r$ are the same, say q, we shall write $\pi(q)$ instead of $\pi(\mathbf{q}_r)$.

Assume that there are n backlogged packets, and all use the same value q_r as retransmission probability. Let $Q_r(i,n)$ be the probability that i out of the n backlogged packets retransmit at the slot. Let $Q_r(1,0) = 0$. Then $Q_r(i,n) = \binom{n}{i}(1-q_r)^{n-i}q_r^i$. Let $Q_a(i,n)$ be the probability that i unbacklogged nodes transmit packets in a given slot (i.e. that i arrivals occurred at nodes without backlogged packets). Let $Q_a(1,m) = 0$. Then $Q_a(i,n) = \binom{m-n}{i}(1-q_a)^{m-n-i}q_a^i$.

2 Team Problem

In this section we propose and analyze three different schemes. We observe that standard slotted Aloha is a special case of these proposed schemes.

Scheme 1: Random power levels without priority: A mobile transmits a packet (new or backlogged) using one of N distinct available power levels uniformly chosen and that does not depend on the type of packet. In case all nodes use the same value q and q_r, the transition probabilities of the Markov chain is given in [2].

Scheme 2: Retransmission with more power: A backlogged packet retransmits with a power from N different distinct levels. A new arrival packet uses a lower power than any one these N levels. The random power levels are chosen uniformly. Successful capture occurs if one of the backlogged packet transmits with a power level larger than that chosen by all others transmitters or if a single new arrival occurs and there is no retransmission attempt of any backlogged packet. The transition matrix is given in [2].

Scheme 3 : Retransmission with less power: A new transmitted packet has the highest power. Backlogged packets attempt retransmissions with a random power choice among N distinct lower power levels. The random power levels are chosen uniformly. The transition matrix is given in [2].

Performance Metrics. Denote by $\pi_n(q)$ the equilibrium probability that the network is in state n and $P(q)$ the transition matrix of a scheme. Then we have the equilibrium state equations:

$$\begin{cases} \pi(q) = \pi(q)P(q), \\ \pi_n(q) \geq 0, n = 0, ..., m \\ \sum_{n=0}^{m} \pi_n(q) = 1. \end{cases} \tag{1}$$

The average number of backlogged packets is

$$S(q) = \sum_{n=0}^{m} \pi_n(q)n. \tag{2}$$

The system throughput (i.e. the sample time average of the number of packets that are successfully transmitted) is given almost surely by the constant,

$$thp(q) = \begin{cases} \sum\limits_{n=1}^{m} \pi_n(q) \Big[Q_a(0,n) \sum\limits_{j=1}^{n} Q_r(j,n)A_j + Q_a(1,n) \sum_{j=0}^{n} Q_r(j,n)A_{j+1} + \\ \qquad \sum\limits_{i=2}^{m-n} Q_a(i,n) \sum_{j=0}^{n} Q_r(j,n)A_{i+j} \Big] + \pi_0(q)Q_a(1,0) & \text{scheme1} \\ \sum\limits_{n=1}^{m} \pi_n(q) \left[Q_a(1,n)Q_r(0,n) + Q_r(1,n) + \right. \\ \qquad \left. \sum_{j=2}^{n} Q_r(j,n)A_j \right] + \pi_0(q)Q_a(1,0) & \text{scheme2} \\ \sum\limits_{n=0}^{m} \pi_n(q) \Big[Q_a(1,n) + Q_r(0,n) \Big\{ Q_r(1,n) + \sum_{j=2}^{n} Q_r(j,n)A_j \Big\} \Big] & \text{scheme3} \end{cases}$$

where A_k is the probability of successful transmission among $k \geq 2$ packets and is given by $A_k = k \sum\limits_{l=0}^{N-1} X_{N-l}(1 - \sum_{i=N-l}^{N} X_i)^{k-1}, A_0 = 0, A_1 = 1$ and X_i is the probability that a packet (new arrival or backlogged) will choose power level i for retransmission. The throughput satisfies

$$thp(q) = q_a \sum_{n=0}^{m} \pi_n(q)(m-n) = q_a(m - S(q)). \tag{3}$$

Eq.(3) equals to the expected number of arrivals per slot (which actually enter the system), and to the expected number of departures per slot. The expected delay of transmitted packets D, is defined as the average time that a packet takes from its source to the receiver. Applying Little's result, this is given by

$$D(q) = 1 + \frac{S(q)}{thp(q)} = 1 + \frac{S(q)}{q_a(m - S(q))} \tag{4}$$

The first term accounts for the first transmission from the source. Combining the last equality in (3) with (4) it follows that maximizing the global throughput is equivalent to minimizing the average delay of transmitted packets. We shall therefore restrict in our numerical investigation to maximization the throughput. However, we shall consider the delay of *backlogged packets* (EDBP) as yet another objective to minimize.

Performance Measures for Backlogged Packets. The throughput of the backlogged packets for each scheme is given by, $thp^c = thp(q) - \Delta$ where Δ is given by:

$$\Delta = \begin{cases} \sum\limits_{n=0}^{m} \sum\limits_{i=1}^{m-n} \sum\limits_{j=0}^{n} \Big\{ \frac{i}{i+j} Q_a(i,n)Q_r(j,n)A_{i+j} \Big\} \pi_n(q) & \text{scheme 1} \\ \sum\limits_{n=0}^{m} Q_a(1,n)Q_r(0,n)\pi_n(q), & \text{scheme 2} \\ \sum\limits_{n=0}^{m} Q_a(1,n)\pi_n(q) & \text{scheme 3} \end{cases}$$

The EDBP D^c is the average time, in slots, that a backlogged packet takes to go from the source to receiver. Applying Little's Theorem, the expected delay of packets that arrive and become backlogged is given by:

$$D^c(q) = 1 + S(q)/thp^c(q) \tag{5}$$

The team problem is therefore given as the solution of

$$\max_q \; objective(q) \; s.t. \begin{cases} \pi(q) = \pi(q)P(q), \\ \pi_n(q) \geq 0, n = 0, ..., m \\ \sum_{n=0}^{m} \pi_n(q) = 1. \end{cases}$$

The solution can be obtained by computing recursively the steady state probabilities, as in Problem 4.1 in [5], and thus obtain an explicit expression for $thp(q)$ as a function of q.

Stability. Slotted aloha is known to have a bi-stable behavior, and we shall check whether this is also the case in our new schemes. We try to find the *drift*, D_n in state n which is the difference between the expected number of new arrivals and new successful departures over a slot time, starting in state n [5].

$$D_n = (m - n)q_a - P_{succ} \tag{6}$$

where the probability of a successful transmission, P_{succ} for each scheme is given in [2]. For standard slotted aloha it has been observed (see [5]) that there are three equilibria, where an equilibrium is defined as a state n in which the arrival rate $(m - n)q_a$ equals the departure rate P_{succ}. Moreover, among those three, the two extreme ones (the one corresponding to the smallest and to the largest state) are stable.[2] A bi-stable situation is undesirable since it could mean that the system spends long time in each of the stable equilibria including in the one with large n (low throughput and large delays).

3 Game Problem

This formulation is of interest in decentralized scenarios where mobiles may not be controlled by a centralized entity. The equilibrium concept then replaces the optimality concept from the team problem. It possesses a robustness property: at equilibrium, no mobile has incentive to deviate.

For a given policy vector $\mathbf{q_r}$ of retransmission probabilities for all users (whose jth entry is q_r^j), define $([\mathbf{q_r}]^{-i}, q_r^i)$ to be a retransmission policy where user j retransmits at a slot with probability q_r^j for all $j \neq i$ and where user i retransmits with probability q_r^i and $[\mathbf{q_r}]^{-i}$ represents the policy vector without user i. Each user i seeks to maximize his own $objective_i(\mathbf{q})$. We then seek for a symmetric

[2] Recall that an equilibrium is stable if the drift corresponding to a small deviation (increasing or decreasing n) from the equilibrium is in the direction opposite to the deviation.

equilibrium policy $\mathbf{q_r^*} = (q_r, q_r, .., q_r)$ such that for any i and any retransmission probability q_r^i,

$$objective_i(\mathbf{q_r^*}) \geq objective_i([\mathbf{q_r^*}]^{-i}, q_r^i) \tag{7}$$

where the objective function is the throughput or minus the expected delay. Since we restrict to symmetric $\mathbf{q_r^*}$, we shall also identify it with the actual transmission probability (which is the same for all users). Next we show how to obtain an equilibrium. We note that due to symmetry, to see whether $\mathbf{q_r^*}$ is an equilibrium it suffices to check (7) for a single player. We shall thus assume that there are $m+1$ users all together, and that the first m users retransmit with a given probability $\mathbf{q_r}^{-(m+1)} = (q^o, .., q^o)$ and user $m+1$ retransmits with probability $q_r^{(m+1)}$. Define the set

$$\mathcal{Q}^{m+1}(\mathbf{q_r}) = \arg \max_{q_r^{(m+1)} \in [\epsilon, 1]} \left(objective_{m+1}([\mathbf{q_r}]^{-(m+1)}, q_r^{(m+1)}) \right)$$

where $\mathbf{q_r}$ denotes the policy where all users retransmit with probability q_r^o, and where the maximization is taken with respect to $q_r^{(m+1)}$. Then q_r^* is a symmetric equilibrium if $q_r^* \in \mathcal{Q}_r^{m+1}(q_r^*)$. To compute the performance measures of interest $objective_{m+1}([\mathbf{q_r}]^{-i}, q_r^i)$, we introduce again a Markov chain with a two dimensional state. The first component corresponds to the number of backlogged packets among users 1,...,m, and the second is the number of backlogged packets (either 1 or 0) of user $m+1$. The various schemes considered are the same as in the team problem.

Performance Metrics. In the game problem, the average number of backlogged packets of source $m+1$ is given by:

$$S_{m+1}([\mathbf{q_r}]^{-(m+1)}, q_r^{m+1}) = \sum_{n=0}^{m} \pi_{n,1}([\mathbf{q_r}]^{-(m+1)}, q_r^{(m+1)}) \tag{8}$$

and the average throughput of user $m+1$ is given by

$$thp_{m+1}([\mathbf{q_r}]^{-(m+1)}, q_r^{(m+1)}) = q_a \sum_{n=0}^{m} \pi_{n,0}([\mathbf{q_r}]^{-(m+1)}, q_r^{(m+1)}) \tag{9}$$

Hence the expected delay of transmitted packets of user $m+1$ is given by

$$D_{m+1}(\mathbf{q_r}]^{-(m+1)}, q_r^{m+1}) = 1 + \frac{S_{m+1}([\mathbf{q_r}]^{-(m+1)} q_r^{m+1})}{thp_{m+1}([\mathbf{q_r}]^{-(m+1)}, q_r^{m+1})} \tag{10}$$

Let us denote by thp_{m+1}^c the throughput of backlogged packets (i.e. of the packets that arrive and become backlogged) at source $m+1$:

$$thp_{m+1}^c(\mathbf{q}_{m+1}) = \sum_{n=0}^{m} \sum_{n'=0}^{m} P_{(n,0),(n',1)}(\mathbf{q}_{m+1}) \pi_{n,0}(\mathbf{q}_{m+1})$$

Thus, the expected delay of backlogged packets at source $m+1$, is given by

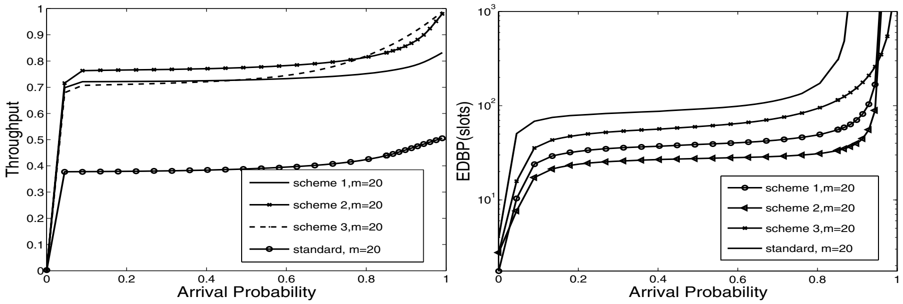
$$D_{m+1}^c(\mathbf{q}_{m+1}) = 1 + S_{m+1}(\mathbf{q}_{m+1}) / thp_{m+1}^c(\mathbf{q}_{m+1}) \tag{11}$$

4 Numerical Investigation

In Figure 1 (a) and (b), we plot the throughput and EDBP for all schemes under the objective of maximizing the global throughput for $m = 20$ and $N = 5$. A general observation is that all new schemes outperform standard Aloha (Figure 1 (a)). The throughput of all the new schemes are comparable under all values of load. The result of maximizing throughput on the EDBP is plotted in Figure 1(b). All the schemes outperform standard Aloha but perform very bad at heavy load. We observe in Fig 1(b) that under low and moderate load, scheme 2 performs the best in minimizing EDBP as scheme 2 prioritizes the backlogged packets. Scheme 1 performs better than scheme 3 by prioritizing a fraction of backlogged packets while scheme 3 gives no priority to backlogged packets. The optimal retransmission probability in scheme 2 is higher than in scheme 1 and standard aloha which explains the best performance scheme 2 in terms of EDBP ([3]) and higher EDBP in scheme 1 and standard aloha.

Inspite of higher optimal retransmission probability in scheme 3 (compared to scheme 1 and standard aloha), scheme 3 experiences higher EDBP compared to scheme 2. In scheme 3, the new transmitted packets have the highest power and higher retransmission probability doesn't affect the successful transmission of new packets implying insensitivity of system's throughput to retransmission probabilities and resulting in dramatic increases in the EDBP and the best performances in the throughput and expected delay of transmitted packets, EDTP. Morever, in heavy traffic or when the number of mobiles increases, the throughput in scheme 3 is exactly the arrival probability: as the system is more congested, and the new arrivals have more priority, the steady probability is then given by $\pi_{m-1} \approx 1$. Hence, only the new packet arrived at free mobile, can be successfully transmitted.

Minimizing EDBP. On maximizing the global throughput we observed a huge EDBP under all schemes in heavy load which may be detrimental to many appli-



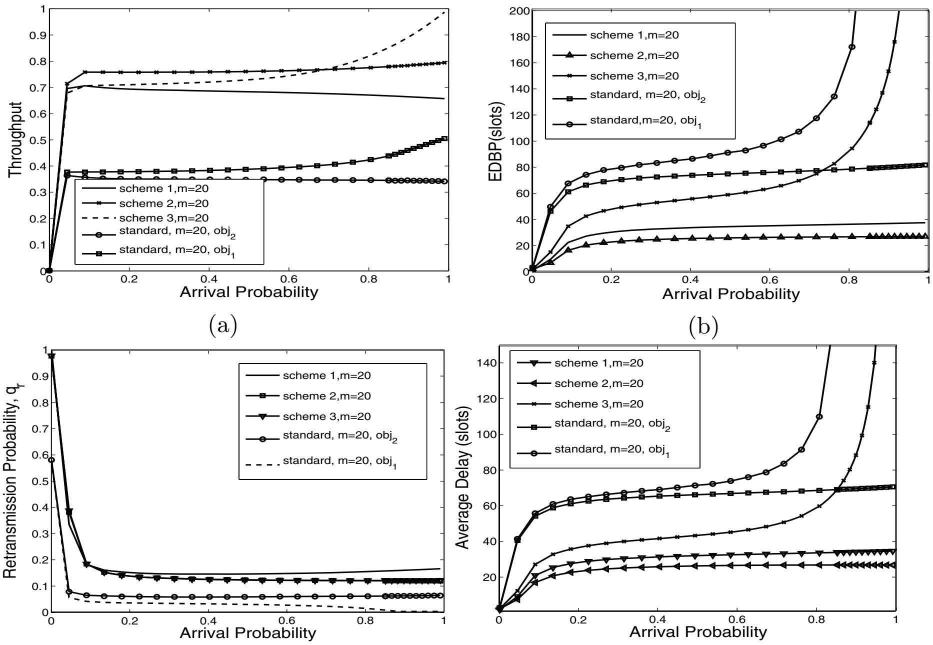
(a) (b)

Fig. 1. (a) and (b) show the throughput and EDBP. The objective under all the schemes is to maximize the throughput. The number of mobiles is 20 and the number of levels is 5 cation. We thus investigate directly the problem of minimizing EDBP and its impact on the throughput performance. We find in particular that throughput performance in the new schemes improves considerably with respect to standard aloha even when standard aloha uses the previous objective of maximizing throughput.

In Figure 2, we plot the performance of the various schemes for $m = 20$. Part (a) considers the impact on the throughput while minimizing the EDBP. Furthermore, all three schemes outperform standard aloha even when the latter uses throughput maximization as optimization objective. We observe that the performance of scheme 3, in contrast to other schemes, is insensitive to the choice of the objective optimization (throughput or EDBP), which explains the high throughput and dramatic increases in the EDBP (see previous paragraph). Thus scheme 3 is very inefficient for many applications using the real time connections. In part (b) of the figure we see that scheme 2 outperforms others in terms of EDBP under all load. Comparing the throughput in scheme 3 and scheme 2, we observe little performance degradation in the throughput of scheme 2, only in heavy traffic. Part (c) provides the optimal retransmission probabilities for standard aloha when the throughput is maximized, which explains its corresponding large EDBP. In contrast, when the EDBP is minimized Aloha has optimal retransmission probabilities of around 0.1 in heavy load whereas all other versions have much higher retransmission probabilities. Part (d) shows the average delay, (EDTP+EDBP)/2 of all the schemes.

An interesting observation that can be made by comparing scheme 2 and other MAC protocols such as CSMA/CA. In contrast to aloha and slotted Aloha, some other MAC protocols such as the CSMA/CA, used in IEEE 802.11, have the feature that a new arriving packet has to wait some (random) amount of time before attempting transmission, so as to avoid a possible collision. In our scheme 2, the impact of new arrivals on the system is similar to have such a random delay, since new arriving packets have less priority (power) and thus on transmitting upon arrival, they cannot cause collisions with a transmitted backlogged packet (if there is one). One can thus view this lower priority as generating a random delay which is, however, more flexible in our scheme since, unlike CSMA/CA, not all arriving packets have to wait a random time. At light load the arriving packets will wait less frequently than in heavy load since the probability of having a competing transmission at the same slot by a backlogged packet is smaller in light traffic.

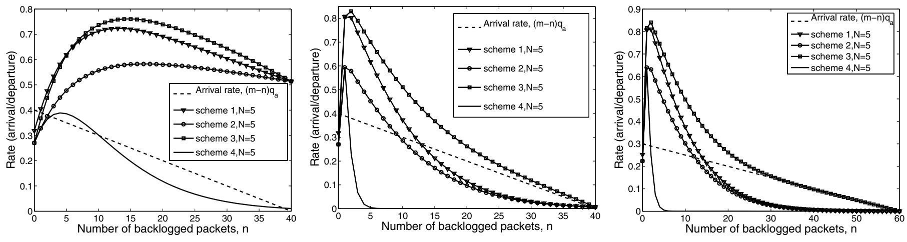
Stability. In Figure 3(a), we illustrate the stability behavior for $q_r = 0.15, q_a = 0.01, m = 40, N = 5$. The drift is the difference between the curves (representing the departure rate or P_{succ}) and the straight line representing the arrival rate $(m - n)q_a$. The system, although fluctuating, tends to move in the direction of the drift and consequently tends to cluster around the two stable points with rate excursions between the two (for scheme 1). Slotted Aloha is the only scheme that suffers from the bi-stability problem and the departure rate is at most $1/e$ whereas for different power schemes it is quite higher. By choosing a large value of retransmission probabilities, we can obtain situations where schemes 1 and 2 acquire a bi-stable regime, and the scheme 3 remains stable for all values



(a) (b)

(c) (d)

Fig. 2. (a), (b), (c) and (d) show the throughput, EDBP, optimal retransmission and the average delay respectively when the objective is to minimize EDBP, obj_2. obj_1 refers to the objective of maximizing the throughput. The number of mobiles is 20



(a) (b) (c)

Fig. 3. Stability and instability of slotted Aloha schemes 1, 2, and 3: (a) $q_a = 0.01$, $m = 40$, $N = 5$ and $q_r = 0.15$. (b) $q_a = 0.01$, $m = 40$, $N = 5$ and $q_r = 0.8$, (c) $q_a = 0.005$, $m = 60$, $N = 5$ and $q_r = 0.8$ for all schemes

of retransmission probability q_r. For example, with 40 mobiles and $q_a = 0.15$, scheme 1 and 2 suffer from the bi-stable problem when $q_r = 0.8$ (see Figure 3 (b)). Note that the bi-stability can occur in all schemes, which is the case when the number of mobiles becomes large. For example, with 60 mobiles and $q_a = 0.005$, the standard slotted Aloha is bi-stable already for $q_r = 0.1$, Scheme 1

Table 1. Average number of backlogged packets (ABP) and equilibrium point (s)

schemes	same-power	no-prior	more-power	less-power
ABP $q_r = 0.1$	56.8	0.71	1.04	1.09
(un)stable eq. $q_r = 0.1$	1.51, 25.47 56.88	0.67	1.449	1
ABP $q_r = 0.5$	60	0.28	0.28	0.287
(un)stable eq. $q_r = 0.5$	1.51, 25.47 56.88	0.136, 28.85 56.83	0.10, 24.33 56.89	0.208
ABP $q_r = 0.9$	60	59.98	59.98	57.58
(un)stable eq. $q_r = 0.9$	0.16, 1.72 60	0.07, 12.43 59.98	0.16, 10.46 59.98	0.11, 26.70 56.99

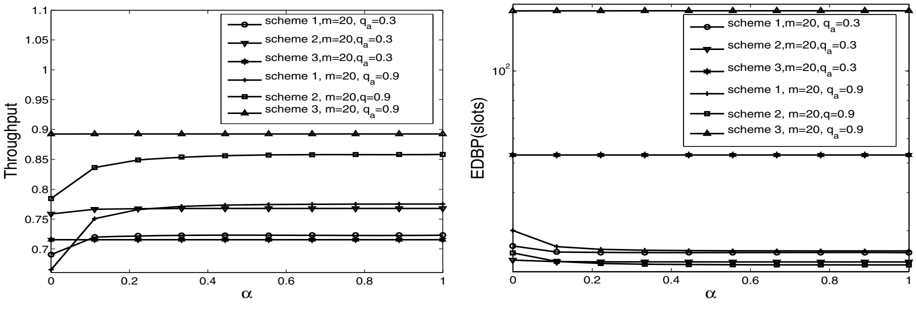
and 2 are bi-stable with $q_r = 0.5$ and Scheme 3 becomes bi-stable with $q_r = 0.95$ (see Fig 3(c)). Here, as well as in all other examples (not reported here) scheme 3 always turned out to have the largest region of parameters for which a unique stable point is obtained.

The average number of backlogged packets for (ABP) different schemes which correspond to their equilibrium points are given in Table 1 with $m = 60$, $q_a = 0.005$ and $N = 5$. In the case of a single equilibrium, a good match is seen for schemes 1, 2 and 3, which means that the simple computation of the stable equilibrium can be used to approximate the expected number of backlogged packets. In standard Aloha we see that the congested stable equilibrium provides a very good approximation for the expected number of backlogged packets, which suggests that the system spends most of the time at that equilibrium.

We also observe same behavior in scheme 1 and 2 when the retransmission probability increases (around 0.5). Scheme 1 and 2 acquire bi-stable equilibrium with $q_r = 0.5$. But contrary to standard aloha, we see from Table 1 that the expected number of backlogged packets for scheme 1 and 2 *can be approximated by the desired stable equilibrium.* Now if the mobiles become aggressive (q_r around 0.9), we see that the congested stable equilibrium provides a very good approximation for the expected number of backlogged packets in all schemes.

Multi-criteria Objective. Consider the objective given by $\alpha thp(q) + (1 - \alpha)\frac{1}{D^c}, 0 \leq \alpha \leq 1$. This allows to handle QoS constraints: By varying α one can find appropriate tradeoff between the throughput and delays, so that the throughput be maximized while keeping the EDBP bounded by some constant. We plot the performance when $N = 5$ and the number of mobiles is 20, for all the new power schemes in Figure 4 (a) and (b). We observe that scheme 3 (less-power for retransmission) is insensitive to the value of α under different load. The optimal retransmission probabilities for scheme 3 are very close under both the objectives: when throughput is maximized and when EDBP is minimized. When α increases (i.e., gives more weight to throughput), the throughput of scheme 2 and scheme 1 increase with α at higher load. But the delay also increases for scheme 2 and scheme 1.

Game Problem: Maximizing individual Throughputs: We evaluate the performance of distributed game version of the problem considering throughput

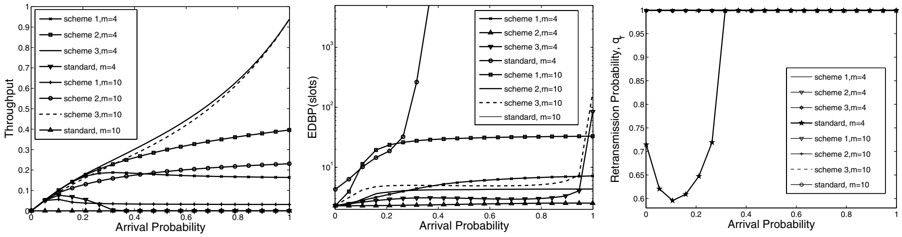


(a) (b)

Fig. 4. (a) and (b) show throughput and delay of backlogged packets when the multi-criteria objective is optimized for $N = 5, m = 20$

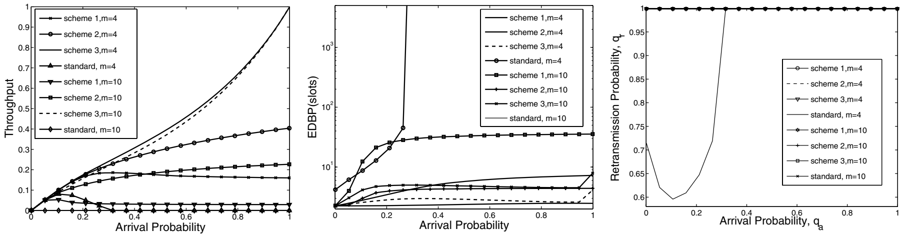
maximization.Fig 5 (a), (b) and (c) show the global equilibrium throughput (i.e. the expression in Eq. (9) times the number of mobiles), the equilibrium EDBP and the equilibrium retransmission probabilities, respectively. We observe that the performance of scheme 2 is the best in terms of EDBP. But at high load, the throughput of scheme 3 is best. The equilibrium retransmission probability is high (close to 1) and thus most aggressive for schemes 1-3 for any arrival probability. Standard aloha is less aggressive at equilibrium at light load. Possible explanation: If an individual tagged mobile were very aggressive in standard aloha (retransmission probabilities close to 1) then eventually all other mobiles would become backlogged which could increase the collision rate and thus decrease the throughput of the tagged mobile. Hence for some values of arrival probabilities the equilibrium behavior of standard aloha is not very aggressive. In contrast, the new schemes suffer less due to other mobiles becoming backlogged since they can reduce collisions due to the randomization and priorities. Hence increasing backlog of other mobiles does not penalize the tagged station anymore, so it has incentive to become more aggressive. The equilibrium transmission probabilities for schemes 1-3 are constants as function of q_a given by 0.97 (for $m = 4, 10$). A remarkable feature of the new schemes is that the equilibrium throughput *increases* with q_a, as in the team problem. In contrast, for high loads the throughput decreases for scheme 1 and it also contains a decreasing behavior in standard Aloha. Thus the competition in the game formulation does not allow to benefit from increased input rates for standard aloha and scheme 1 (except for low values of q_a) whereas the new schemes do benefit from that. Finally, we note that schemes 1-3 all avoid the throughput collapse of standard aloha (for which we see in Fig. 5 (a) that the equilibrium throughput vanishes for both $m = 10$ at $q_a > 0.3$ and for $m = 10$ at $q_a > 0$).

Game Problem: Minimizing Individual EDBP. Next, we evaluate the performance of the distributed game problems of minimizing EDBP. We notice again from Figure 6(a) that the equilibrium throughput decreases with q_a for scheme



(a) (b) (c)

Fig. 5. (a), (b) and (c) show the throughput, EDBP and retransmission probability when the objective is to maximize the throughput for all the schemes for 4 and 10 mobiles and the number of power levels is 5



(a) (b) (c)

Fig. 6. (a), (b), and (c) show the throughput, EDBP and retransmission probability when the objective is to minimize the delay of backlogged packets for all the schemes for 4 and 10 mobiles and the number of levels is 5

1 (for arrival probabilities larger than 0.2) and for standard aloha (for $q_a > 0$). In both schemes 2 and 3 it increases yet the increase is much larger in scheme 3. This scheme outperforms all others for any q_a. Schemes 1-3 all avoid the throughput collapse of standard aloha. We observe a non-monotonic behavior of the equilibrium EDBP for scheme 3 in Figs 6(b). According to Eq. (11), this means that as the arrival rate increases, the throughput grows faster than the expected number of backlogged packets. Scheme 2 and 3 have very close EDBP which is better than scheme 1 and standard aloha for all q_a. we see that 1-3 are very aggressive in terms of retransmission probabilities in Figs 6(c). An interesting feature to note is that the throughput obtained when maximizing the individual throughput is less than that obtained when minimizing the EDBP. This is due to the fact that we are in a non-cooperative game setting, for which the equilibria are known not to be efficient (as is the case in the famous prisoner's dilemma paradox).

5 Conclusions

We have studied two schemes that involve both prioritization as well as power diversity for increasing the throughput and decreasing the EDBP. We studied optimal choices of transmission probabilities both in a cooperative as well as in a non-cooperative setting. Scheme 3 has the best stability properties and the best throughput performance in the game setting. The throughput performance of schemes 2 and 3 benefit from increasing the arrival rate in the game scenario, in contrast with standard Aloha which suffers a throughput collapse, and with the power diversity scheme 1 (without priorities) whose equilibrium throughput decreases in high load. A remarkable feature of scheme 3 is that it performs very well in the game setting as compared to the team problem. In particular, when maximizing the throughput, we see that in heavy traffic it attains the maximum achievable throughput as is the case for the team formulation.

References

1. N. Abramson, "The Aloha system – another alternative for computer communications", *AFIPS Conference Proceedings*, Vol. 36, pp. 295-298, 1970.
2. E. Altman, D. Barman, A. Benslimane and R. El Azouzi, "Slotted Aloha with priorities and random power", `http://www.inria.fr/mistral/personnel/Eitan.Altman/mobile.html`
3. E. Altman, D. Barman, R El Azouzi and T. Jimenez, "A game theoretic approach for delay minimization in slotted aloha", ICC, 20-24, Paris, France, June 2004.
4. E. Altman, R El Azouzi and T. Jimenez, "Slotted Aloha as a Stochastic Game with Partial Information", WiOpt'03, Sophia Antipolis, France, March 3-5, 2003.
5. D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, New Jersey, 1987.
6. Y. Jin and G. Kesidis, "Equilibria of a noncooperative game for heterogeneous users of an ALOHA network", *IEEE Comm. Letters* **6** (7), 282-284, 2002.
7. R. O. LaMaire, A. Krishna and M. Zorzi, "On the randomization of transmitter power levels to increase throughput in multiple access radio systems", *Wireless Networks* **4**, pp 263–277, 1998.
8. A. B. MacKenzie and S. B. Wicker, "Selfish users in Aloha: A game theoretic approach", *IEEE VTC*, fall, 2001.
9. A. B. MacKenzie and S. B. Wicker, "Stability of Slotted Aloha with Multipacket Reception and Selfish Users," Infocom, April 2003.
10. J. J Metzner, On improving utilization in ALOHA networks, IEEE Transaction on Communication COM-24 (4), 1976.
11. L. G. Roberts, "Aloha packet system with and without slots and capture", *Tech. Rep. Ass Note 8*, Stanford Research Institute, Advance Research Projects Agency, Network Information Center, 1972.
12. J. H. Sarker, M. Hassan, S. Halme, Power level selection schemes to improve throughput and stability of slotted ALOHA under heavy load, Computer Communication 25, 2002.
13. Yalin E. Sagduyu and Anthony Ephremides, "Power Control and Rate Adaptation as Stochasic Games for Random Access", Proc. 42nd IEEE Conference on Decision and Control, Hawaii, Dec. 2003

Downlink Packet Scheduling with Minimum Throughput Guarantee in TDD-OFDMA Cellular Network

Young Min Ki, Eun Sun Kim, Sung Il Woo, and Dong Ku Kim

Yonsei University, Dept. of Electrical and Electronic Engineering,
134 Shinchon-Dong, Seodaemun-Gu, Seoul 120-749, Korea
{mellow, esunkim, niceguy8189, dkkim}@yonsei.ac.kr
http://mcl.yonsei.ac.kr

Abstract. This paper proposes a channel scheduler in downlink of IEEE 802.16e TDD-OFDMA cellular network which provides users the minimum throughput for best effort traffic. The proposed scheduler is further developed in the consideration of the case that network might become unfeasible, where extra weight functions are introduced into the proportional fair (PF) scheduler so that the proposed extra weight function provides more resources to users who are easier to achieve the target throughput than those who cannot. It does not guarantee target throughput for all users but eventually increases the number of users being served with the guaranteed minimum throughput. It is investigated for the proposed schedulers and other conventional ones how large the throughput is and how many users are guaranteed the target throughput. Simulation results show that the proposed algorithms have 10 to 20 % more users being served with the minimum guaranteed throughput. The trade-off between the throughput guarantee level and sector throughput is shown in the proposed schedulers.

1 Introduction

Emerging WLANs and future wireless mobile systems are expected to adopt a multi-carrier scheme (OFDM) consisting of hundreds of carriers [1]. For a broadband channel, fading over frequency as well as time should be handled more carefully in order to be able to enhance overall performance. There are two strategies applied to combat frequency variations: "frequency diversity" and "frequency selectivity" [2]. Frequency diversity, which employs the identical modulation and coding schemes over all subchannels, allows system performance not to be dominated by a few deeply faded subcarriers.

The frequency selective strategy exploits channel information over frequency to achieve adaptation gain, which is realized by the use of different link adaptations and channel schedulers in each subchannel. The studies of making the OFDMA more efficient in terms of frequency domain scheduling has been addressed in the number of papers [2-7]. However, it is still unclear what scheduling algorithm maximizes the sector throughput while still providing users with

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 623–633, 2005.

certain QoS such as minimum throughput or delay requirement. Such QoS guarantees would ensure that the network renders the service to all the users. In order to highlight this paradigm, the present paper evaluates the performance of scheduling algorithms under minimum user throughput guarantees in the IEEE 802.16e TDD-OFDMA [8-10] downlink network.

2 Packet Scheduling for TDD-OFDMA Downlink

2.1 Downlink Scheduling for TDD-OFDMA

The OFDMA is one of the time and frequency division multiple access techniques based on the OFDM and is currently used in the IEEE 802.16d [8] and 802.16e [9-10] standards. The fixed length OFDMA frame consists of a block of downlink (DL) OFDM symbols followed by another block of (UL) OFDM symbols. In the frequency domain, full RF bandwidth is divided into hundreds of subcarriers. Numbers of subcarriers are bunched into the subchannel, which can be handled as a minimal resource unit. The time division duplexing (TDD) does not provide either continuous downstream or continuous upstream. The channel feedback period is determined to be the same as OFDMA frame duration. Therefore, scheduling should be able to be performed at most every frame. Scheduling in OFDM/TDM allocates whole bands to only one user while OFDMA can assign different users over different subchannels. Fig. 1 shows an example of channel scheduling that happens in TDD-OFDMA where single OFDMA scheduler is capable to assign different users to different subchannels [2][5].

2.2 Conventional Scheduling Methods

1) *Maximum C/I* (*Max C/I*): One simple method is to serve the user terminal (UT) of index i_n^* at the n-th subchannel for every scheduling instance t with respect to:

$$i_n^* = \arg\max_i R_{i,n}(t), \tag{1}$$

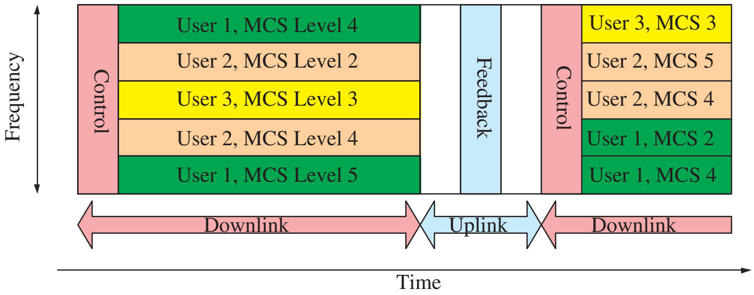


Fig. 1. Priority-based channel scheduling example in TDD-OFDMA downlink

where $R_{i,n}(t)$ denotes the instantaneous supportable data rate of the n-th subchannel of the user i. This serving principle has obvious benefits in terms of system throughput. However, it does not take into consideration of throughput fairness of users, leaving those users of poor average radio conditions served less frequently.

2) *Proportional Fair* (*PF*): To remedy the fairness problem, the proportional fairness (PF) scheduling algorithm was proposed [11-12]. According to the PF scheduling algorithm, the selected UT of index i_n^* should be denoted such that:

$$i_n^* = \arg\max_i \frac{\left(R_{i,n}(t)\right)^{\alpha}}{\left(T_{i,n}(t)\right)^{\beta}}, \tag{2}$$

where $T_{i,n}(t)$ denotes the average throughput of the n-th subchannel of UT i. α and β are indices used to control the scheduling fairness, which are normally set to $\alpha = \beta = 1$. The PF algorithm intends to serve those users seeing very favorable instantaneous radio channel conditions relative to their average ones, thus taking advantage of the temporal variations of the fast fading channel.

3) *Fast Fair Throughput* (*FFTH*): In [13], the fast fair throughput (FFTH) scheduler was suggested that the selected UT of index i_n^* should be defined that:

$$i_n^* = \arg\max_i \frac{R_{i,n}(t)}{T_{i,n}(t)} \cdot \left[\frac{max_j\left(\overline{R_{j,n}(t)}\right)}{\overline{R_{i,n}(t)}}\right], \tag{3}$$

where $R_{i,n}(t)/T_{i,n}(t)$ is a well-known PF factor and $\overline{R_{i,n}(t)}$ is the average supportable data rate of the n-th subchannel of the user i. The expression $max_j\left(\overline{R_{j,n}(t)}\right)$ is a constant that indicates the maximum average supportable data rate from all the users. Note that the $\overline{R_{i,n}(t)}$ term in the denominator is to compensate the priority of less favorable users and distribute evenly the cell throughput to all users.

4) *Fair Throughput* (*FTH*): For throughput fairness in wireless, a simplemethod is considered to serve the user terminal (UT) of index i_n^* at the n-th subchannel for every scheduling instance t with respect to:

$$i_n^* = \arg\max_i \frac{1}{T_{i,n}(t)}, \tag{4}$$

where $T_{i,n}(t)$ denotes the average throughput of the n-th subchannel of UT i. The FTH scheduling is considered as a slow scheduling due to the fact that it does not require any instantaneous information of the channel quality [13].

3 Schedulers for Minimum Throughput Guarantee

3.1 User Throughput Outage

If no constraint were imposed on users' QoS, the Max C/I scheduler would provide maximum system throughput. The fair scheduling algorithms such as PF, FFTH dent and FTH represent the trade-off between throughput and fairness, but cannot provide any QoS guarantees such as minimal throughput. For highly loaded conditions, users having poor average radio channel conditions are allocated very little time resources, starved for throughput, and will ultimately be dropped out of the network without satisfying their requirements [5][13]. A set of user throughput is said to be feasible if the sum of all the user throughputs is lower or equal to the air interface capacity. If the network is feasible, there are certain schedulers that can guarantee all the requirements. In order to be able to guarantee the user throughput requirements, the network should be able to provide a minimum throughput to all users with a certain outage levels allowed in the network. The throughput outage probability of the user i is defined as the probability that user throughput cannot satisfy the required target throughput which is expressed as follows:

$$Pr_i(T_i < T_{i,req}) \leq \delta_i, \tag{5}$$

where T_i is the average throughput of user i and $T_{i,req}$ is the required minimum throughput of the user i that can be determined by user QoS class. δ_i is the required target throughput outage probability of user i.

3.2 Extra Weight Method for Minimum Throughput Guarantee

In [14], various modifications of the PF algorithm were proposed to guarantee the delay requirements. In this paper, the modification of PF algorithm of [14] is introduced for the minimum user throughput guarantees. The selected UT of i_n^* of our modified algorithm is denoted such that:

$$i_n^* = \arg\max_i \begin{cases} \frac{\left(R_{i,n}(t)\right)^\alpha}{\left(T_{i,n}(t)\right)^\beta} \cdot C_{i,n}, & T_i < T_{i,req} \text{ or } Pr_i(T_i < T_{i,req}) > \delta_i \\ \frac{\left(R_{i,n}(t)\right)^\alpha}{\left(T_{i,n}(t)\right)^\beta}, & T_i \geq T_{i,req} \text{ and } Pr_i(T_i < T_{i,req}) \leq \delta_i \end{cases}, \tag{6}$$

where $\left(R_{i,n}(t)\right)^\alpha / \left(T_{i,n}(t)\right)^\beta$ is the well-known PF factor and $C_{i,n}$ is extra weight of the n-th subchannel of user i. $Pr_i(T_i < T_{i,req})$ is the throughput outage probability and Pt_t is target outage of user i. Only needy users are given extra weight $C_{i,n}$ once either user throughput turns out to be unable to guarantee the minimum user throughput or the throughput outage exceeds a certain target level. When there is no throughput violations or $C_{i,n} = 1$, this scheduling algorithms becomes to be equivalent to a conventional PF algorithm. The extra weight $C_{i,n}$ is given as (7):

$$C_{i,n} = \frac{max_j\left(\overline{R_{j,n}(t)}\right)}{\overline{R_{i,n}(t)}}, \tag{7}$$

which is the same form as the weighting function of FFTH of (3). $C_{i,n}$, which is the ratio of the maximum supportable data rate of the system to average supportable data rate of needy user, would allocate more resources to those users who have inferior throughput outage probability.

3.3 Minimum User Throughput in Unfeasible Networks

Since the supportable data rates of users are determined by channel condition in the rate controlled scheduling, the users in the cell edge, who are likely to have lower SINR than one which is required for the transmission of minimum data rate, cannot be served so much as to meet the minimum user throughput. In this case, the increase in priority of the users in this unfeasible area cannot guarantee the minimum user throughput but only decrease system throughput. Since $C_{i,n}$ of (7) would have given too much extra priority to poor channel condition users in unfeasible conditions, it is modified as follows:

$$C_{i,n} = \left[\frac{max_j \left(\overline{R_{j,n}(t)} \right)}{\overline{R_{i,n}(t)}} \right]^{min(1.0, T_i/T_{i,req})}, \tag{8}$$

where $T_i/T_{i,req}$ is the normalized throughput. If the user throughput is close to the required target throughput, the exponent $min(1.0, T_i/T_{i,req})$ becomes to be close to 1.0. If the user throughput becomes smaller, the exponent goes to 0.0. Therefore, the extra priority of (8) cannot guarantee all of minimum user target throughputs. The extra priority of (8) is to provide those users who are easier to achieve the target throughput with more service than those who cannot.

4 Simulation Results

4.1 Simulation Environments

Table 1 shows the IEEE 802.16e based OFDMA parameters [8-10] and the link adaptation table (MCS table) is set to the parameters displayed in Table 2 [15-16]. The MCS level is reported to mobiles according to SINR sensitivity thresholds and one frame delay is assumed in MCS feedback. It is assumed that there are 19 cells with 3 sectors of same frequency allocation (FA) and cell radius is 1 km. User terminals are distributed in the center sector and their locations are generated more than 1,000 times. Only best-effort traffic with full-buffering is considered. The channel C/I of the n-th subchannel of user i can be expressed as follows:

$$(C/I)_{i,n} = \sum_{j=1}^{J} \|\gamma_{j,n}\|^2 \cdot \left(G_i^{-1} + \sum_{k=1}^{K} \|\psi_{k,n}\|^2 \right)^{-1}, \tag{9}$$

where G_i is the average geometry which is determined by path loss and shadowing and shown as

$$G_i = \frac{I_{or}}{I_{oc} + N_0 W} = \frac{1}{I_{oc}/I_{or} + 1/(I_{or}/N_0 W)}, \tag{10}$$

where I_{or} is the received serving-cell pilot strength, I_{oc} is the sum of the received other-cell pilot strength, and $N_0 W$ is the thermal noise power. The expression $\{\gamma_j\}$ represents the multi-path component within the guard interval

Table 1. OFDMA parameters

Parameters	Value
Carrier Frequency	2.3 GHz
Channel Bandwidth	10 MHz
Number of used subcarriers	1,702 of 2,048
Number of traffic subcarriers	1,536
Subcarrier spacing	5.57617 kHz
Number of subchannels	32
Number of subcarriers	48
Frame length	5.0 msec
Number of symbols per frame	26
Number of DL / UL symbols	18 / 8
Sum of RTG and TTG	45.885 μsec
OFDMA symbol time	190.543 μsec
Cyclic prefix	1/16

Table 2. Modulation and coding (MCS) table

Modulation	Code rate	Sensitivity threshold S/N	PHY bit/sec/Hz
QPSK	1/12	-3.0 dB	0.135
QPSK	1/8	-1.3 dB	0.202
QPSK	1/4	1.4 dB	0.404
QPSK	1/2	6.6 dB	0.807
QPSK	3/4	8.5 dB	1.211
16QAM	1/2	10.5 dB	1.613
64QAM	2/3	15.3 dB	3.227
64QAM	3/4	20.8 dB	3.63

and $\{\psi_k\}$ is multi-path component which exceeds the guard interval. In simulations, $\sum_{j=1}^{J} \|\gamma_{j,n}\|^2$ is assumed to be Rayleigh fading and $\{\psi_k\}$ is ignored since the cyclic prefix is assumed to be sufficiently longer than the overall delay spread [12]. The path loss model is assumed to be a vehicular model $129.427 + 37.6 * log_{10}(d_{km})$. The standard deviation of Log-normal shadowing is assumed to be 10 dB. Short-term channel gains are assumed to be Rayleigh fading with a Doppler frequency of 6.4Hz (3km/Hr). The BS transmit power is set to 20 W (43 dBm). Thermal noise density is assumed to be -174 dBm/Hz and max C/I limit is set to 30 dB.

4.2 Sector Throughput Performance

Fig. 2 represents the users' average geometry distribution. It is shown that around 45 % of samples has average geometry inferior to the lowest level which is -3.0 dB as shown in Table 2. In this paper, the interference management tech-

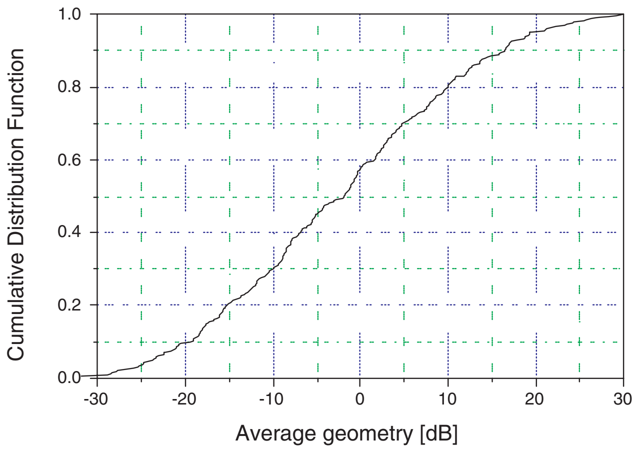


Fig. 2. Users' average geometry distribution

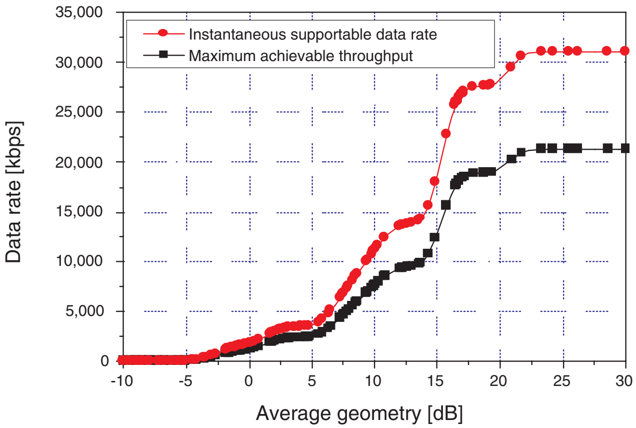


Fig. 3. Instantaneous supportable data rate and maximum achievable throughput according to average geometry

niques such as frequency hopping (FH) or handoff subchannel [9-10] are not considered and system could have trouble of being unfeasible due to those users in the poor channel conditions. Fig. 3 shows the instantaneous supportable data rate and maximum achievable throughput according to average geometry. Since TDD-OFDMA downlink transmits on DL subframe which uses 18 symbols out of 26 symbols, the maximum achievable throughput is around 70 % of instantaneous supportable data rate in parameters of Table 1. Fig. 4 represents sector throughput performance of different scheduling methods for the various number of simulated users. Max C/I scheduler shows the largest multi user diversity gains as number of users increases. FTH and FFTH schedulers attempt to allo-

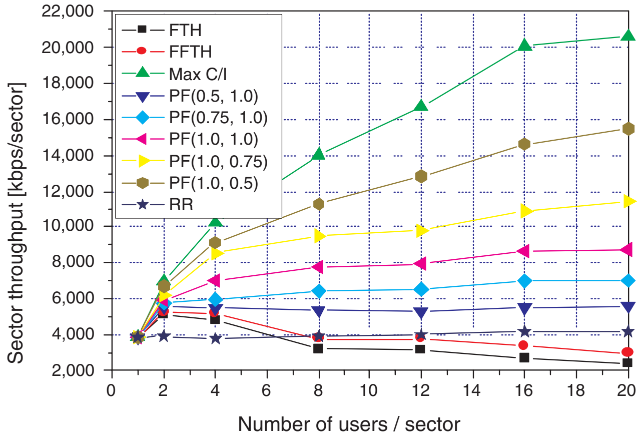


Fig. 4. Sector throughput results of different scheduling methods according to number of simulated users (fairness indices are presented as PF (α, β) in the graph)

cate more resources to those lower throughput users, so provide smaller sector throughput as number of users increase. PF scheduler gives more emphasis on channel factor ($\alpha > \beta$), so more multi user diversity gains are achieved.

4.3 Minimum User Throughput Guarantee Performance

In order to evaluate the throughput guarantee performance of different schedulers, three minimum target throughputs are considered: 64, 128, and 384 kbps. Target throughput outage is set to 5 % for all three cases. Two versions of proposed algorithm of (6) are evaluated: the proposed Algorithm 1 has extra function of (7) and Algorithm 2 adopts extra function of (8). Fairness indices of proposed algorithms are set to $\alpha = \beta = 1$. It was evaluated how many users in average are guaranteed their target throughput and how large is their average throughput. Table 3 shows the minimum throughput guarantee results when the number of simulated users is set to 16. For 64 kbps guarantee, all scheduling algorithms except max C/I provide similar number of guaranteed users but PF and proposed algorithms provide around 2.5 to 3 times more throughput than FTH, FFTH and RR. In this case, the throughput requirement is not tight and all schedulers show good performance. For 128 kbps case, the proposed Algorithm 2 shows the most guaranteed users but show around 30 % throughput loss than PF, which means that the proposed algorithm allocates more resources to throughput violated users than PF. In order to test our algorithms in high loaded unfeasible network, we increase target throughput up to 384 kbps. The FTH, FFTH and Algorithm 1 turns out to give too much resources to those users in poor channel conditions, so it could not guarantee the minimum user throughput and only decrease system throughput. However, the proposed Algorithm 2 guarantees around 25 % more users while decreasing system throughput.

Table 3. Minimum user throughput guarantees of 64, 128, and 384 kbps throughput with 5 % target outage when the number of simulated users is 16

Target outage	Scheduling Algorithm	Average number of guaranteed users	Average throughput of guaranteed users [kbps]	Total throughput of guaranteed users [kbps]
64 kbps (5 %)	Max C/I	1.49	13,456.39	20,050.02
	RR	8.05	495.10	3,985.56
	FTH	9.53	281.53	2,682.98
	FFTH	9.43	355.55	3,352.84
	PF	9.21	935.40	8,615.30
	Algorithm 1	9.55	895.34	8,550.50
	Algorithm 2	9.67	828.21	8,008.79
128 kbps (5 %)	Max C/I	1.43	13,996.38	20,014.82
	RR	6.20	638.48	3,958.58
	FTH	9.25	286.21	2,647.44
	FFTH	8.78	373.10	3,275.82
	PF	8.11	1,048.19	8,500.82
	Algorithm 1	8.84	823.86	7,282.92
	Algorithm 2	9.29	736.18	6,389.11
384 kbps (5 %)	Max C/I	1.27	15,746.70	19,998.31
	RR	4.10	928.94	3,808.65
	FTH	0.83	565.76	469.58
	FFTH	2.65	556.68	1,475.21
	PF	5.34	1,465.90	7,827.91
	Algorithm 1	2.64	644.41	1,701.24
	Algorithm 2	6.74	467.63	3,151.83

4.4 Geographical Fairness Performance

A Ring is defined as the area occupied by a 100 meter unit. For example, the r-th Ring is the area which falls into between the (r-1) hundreds meters and the r hundred meters from the BS. "Average user throughput of the r-th Ring" represents a user average rate in the r-th Ring and is determined as:

$$R_{user}(r) = \frac{R_{ring}(r)}{\text{Number of User in } r\text{-th Ring}}, \tag{11}$$

where $R_{ring}(r)$ is "Ring throughput" of the r-th Ring and is defined as:

$$R_{ring}(r) = \sum_{i \in r\text{-th Ring}} T_i = R_{user}(r) \cdot N_{user}(r), \tag{12}$$

where T_i denotes average throughput of the i-th user and $N_{user}(r)$ denotes number of users in of the r-th Ring. Fig. 5 represents average user throughput per Ring in PF and proposed algorithms when the number of simulated users is 16.

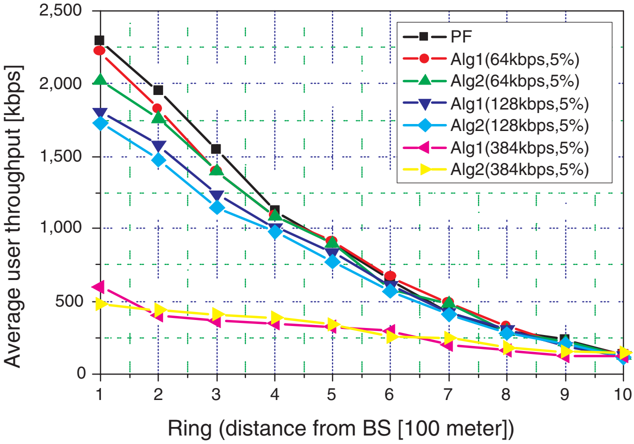


Fig. 5. Average user throughput per Ring in PF and proposed scheduling algorithms when the number of simulated users is 16

In 64 and 128 kbps target throughput guarantee cases, Algorithm 1 shows better throughput performance than Algorithm 2, but Algorithm 2 provides more guaranteed users. However, for 384 kbps case, Algorithm 1 outperforms to Algorithm 2 only in the first Ring. It is also shown that the user throughput becomes more geographically fair for higher target throughput, resulting in total throughput decrease.

5 Concluding Remarks

The total throughput and minimum throughput guarantee performance of best effort service were evaluated for downlink of an IEEE 802.16e TDD-OFDMA cellular network. The conventional schedulers without minimum throughput guarantee shows trade-off between throughput and fairness. To be able to guarantee the minimum user target throughput, the extra weight functions were introduced into the proportional fair (PF) scheduler. The proposed extra weight algorithms provide users who are easier to achieve the target throughput with more service than those who cannot, among those users not satisfying the target throughput due to the poor channel conditions that might also make system become at the risk of being unfeasible. The proposed algorithm does not guarantee target throughput for all users but eventually increase the number of the guaranteed users. Simulation results show that the proposed algorithms represent 10 to 20 % more users who can be served with the minimum guaranteed throughput.

Acknowledgments

This work was supported by the Korea Research Foundation Grant (KRF-2004-013-D00060). Also, this work was supported by grant No. (R01-2002-000-00531-0) from the Basic Research Program of the Korea Science & Engineering Foundation.

References

1. Shakkottai S., Rappaport T. S.: Cross-Layer Design for Wireless Networks. Communications Magazine, Vol. 41. IEEE. (2003) 74-80
2. Classon B., Sartori P., Nangia V., Zhuang X., Baum K.: Multi-dimensional Adaptation and Multi-user Scheduling Techniques for Wireless OFDM Systems. IEEE International Conference on Communications (ICC '03), (2003)
3. Anchun W., Liang X., Shidong Z., Xibin X., Yan Y.: Dynamic Resource Management in the Forth Generation Wireless Systems. International Conference on Communication Technology (ICCT2003), (2003)
4. Xiao L., Wang A., Zhou S., Yao Y.: A Dynamic Resource Scheduling Algorithm for OFDM System. The 9th Asia-Pacific Conference on Communications, 2003 (APCC 2003)
5. Ki Y. M., Kim E. S., Kim D. K.: Downlink Scheduling and Resource Management for Best Effort Service in TDD-OFDMA Cellular Networks. Lecture Notes in Computer Science, Vol. 3260. (2004) 315-329
6. Das S., Viswanathan H.: Dynamic Power and Sub-carrier Assignment in a Multiuser OFDM System. IEEE 60th Vehicular Technology Conference (VTC 2004)
7. Kim K., Kim H., Han Y.: Subcarrier and Power Allocation in OFDMA Systmes. IEEE 60th Vehicular Technology Conference (VTC 2004)
8. http://www.ieee802.org/16/tgd/
9. http://www.ieee802.org/16/tge/
10. IEEE P802.16e/D3, Draft Amendment to IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems - Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, (2004)
11. Jalali A., Padovani R., Pankaj P.: Data throughput of CDMA-HDR a high efficiency-high data rate personal communication wireless system. The 51st IEEE Vehicular Technology Conference, 2000 (VTC 2000-Spring)
12. 3GPP R1-030042, Update of OFDM SI simulation methodology. (2003)
13. Ameigeiras P., Wigard J., Mogensen P.: Performance of Packet Scheduling Methods with Different Degree of Fairness in HSDPA. IEEE 60th Vehicular Technology Conference (VTC 2004)
14. Barriac G., Holtzman J.: Introducing Delay Sensitivity into the Proportional Fair Algorithm for CDMA Downlink Scheduling. IEEE Seventh International Symposium on Spread Spectrum Techniques and Applications (2002)
15. IEEE C802.16e-03/22r1, Convergence simulations for OFDMA PHY mode, (2003)
16. IEEE C802.16d-03/78r1, Coverage/Capacity simulation for OFDMA PHY in with ITU-T channel model, (2003)

Making IGP Routing Robust to Link Failures

Ashwin Sridharan[1] and Roch Guérin[2]

[1] Sprint ATL, 1 Adrian Court, Burlingame CA 94010
ashwin@sprintlabs.com
[2] University of Pennsylvania
guerin@ee.upenn.edu

Abstract. An important requirement of a robust traffic engineering solution is insensitivity to changes, be they in the form of traffic fluctuations or changes in the network topology because of link failures. In this paper we focus on developing a fast and effective technique to compute traffic engineering solutions for Interior Gateway Protocol (IGPs) environments that are robust to link failures in the logical topology. The routing and packet forwarding decisions for IGPs is primarily governed by link weights. Our focus is on computing a *single* set of link weights for a traffic engineering instance that performs well over all *single* logical link failures. Such types of failures, although usually not long lasting, of the order of tens of minutes, can occur with high enough frequency, of the order of several a day, to significantly affect network performance. The relatively short duration of such failures coupled with issues of computational complexity and convergence time due to the size of current day networks discourage adaptive reactions to such events. Consequently, it is desirable to *a priori* compute a routing solution that performs well in all such scenarios. Through computational evaluations we demonstrate that our technique yields link weights that perform well over all *single* link failures and also scales well, in terms of computational complexity, with the size of the network.

Keywords: Routing, Traffic Engineering, IGP, Optimization, link failures.

1 Introduction

With IP networks carrying increasing amounts of commercial traffic, reliability and steady performances have become critical factors. Large ISPs today offer Service Level Agreements (SLAs) covering delay, port availability and losses across their network, which must be maintained even if the network experiences congestion or failures. This provides strong incentives for engineering the network to maintain acceptable levels of performance across a broad range of operating conditions, i.e., fluctuating traffic patterns and intensities or topological changes caused by link and node failures. The focus of this paper is on one such aspect of traffic engineering, namely computation of robust routes to mitigate the impact of single link failures in logical topologies.

Most large service providers map a logical (IP) topology onto an underlying physical fiber network and then operate an Interior Gateway routing protocol like OSPF, IS-IS, or EIGRP on the logical topology. Hence, disruption can occur due to a failure in

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 634–646, 2005.

the physical network, for example a fiber cut, or due to a failure in the logical topology, for example a loss of adjacency between routers due to CPU overload, lost packets, interface failure, etc. Critical failures in the physical network, e.g., fiber cuts, are relatively infrequent but typically long lasting, from hours to days, [8]. As a result, they are dealt with either directly at the physical layer, e.g. automatic switch-over, or with *reactive routing* which re-computes a new routing pattern. The relatively long recovery time of reactive routing is warranted by the scope and severity of such failures. In contrast, logical link failures are much more common but normally affect just a single link (see again [8]), and are typically transient in nature, resolving over a time scale of minutes. Because of their relatively short durations, logical failures are a natural candidate for proactive solutions that rely on *robust routing* which *apriori* anticipate all possible logical link failure scenarios. In particular, relying on reactive solutions may not be appropriate or even feasible, as quickly computing and deploying a complete set of new routes can be challenging, especially in today's large networks that often include several hundred routers. Indeed, by the time the network has determined how to best adapt its routing and propagated the required changes, the failure may have been already fixed or be in the process of being fixed. In addition, re-computing a new optimal routing pattern has the potential for being disruptive to more flows, i.e., re-route them, than just the flows that were originally affected by the failed link.

In this paper, our focus is on developing a technique to compute such *a priori* robust routing solutions for OSPF [9], IS-IS [3] and EIGRP [1], three of the most common intra-domain routing protocols in the Internet. The IGP routing framework imposes two unique constraints. One, traffic in an IGP environment can flow only along shortest paths computed using a set of link weights[1] (say W). Two, load balancing of traffic between a node-pair is allowed only over equal cost multiple paths with the further requirement that traffic be split in *equal* proportions across paths[2]. Within the framework of robust routing, our goal is then to compute a *single* set of link weights for a *given* characterization of traffic entering the network, i.e., traffic matrix T, which performs well under both normal conditions and in the event of any single logical link failures This would ensure that only the routes traversing the failed link are affected and possibly re-routed[3], which is clearly the minimum re-routing required under a link failure. Another important goal is to ensure that the weight computation algorithm scales well with network size which, as mentioned previously, easily span hundreds of routers.

Most of the previous works, on optimizing IGPs for traffic engineering, e.g. [6], [4], [12] and [13] only address the problem in an environment without link failures. As one can expect and as we shall see, performance can degrade significantly if link weights are chosen without accounting for failures. [6] proposed a technique for coping with link failures in OSPF/IS-IS networks, but that involves changing a few link weights upon failure and hence is suitable only for *reactive* routing.

[1] As routing is fully specified by the set of link weights, in this paper we use interchangeably the terms "routing solution" and "weight setting."

[2] Although EIGRP allows unequal splitting, it is rarely used.

[3] Re-routing would not occur if there were multiple shortest paths.

The problem of computing a *single* set of link weights robust to all single link failures was first addressed in [2] and has been studied in [10], [7], and [14]. [10] presents a tabu-search heuristic that computes a set of weights robust to link failures by evaluating the impact of *all* possible link failures for each sampled weight setting. Although it yields good link weights, evaluation of all link failures for a weight setting has very high computation complexity making the search intractable for large networks. In order to avoid this bottleneck a typical method is to evaluate the impact of failure of only a few important links. The key issue then is the identification of such links. [14] relies on random sampling, but this obviously ignores the coupling that exists between links, traffic, and routing. [7] defines important links based on the increase in cost they contribute upon failure. However, the cost increase is computed for the current weight setting, which introduces a strong bias based on the routing that this weight setting induces. In other words, neither the technique of [14] nor that of [7] selects important links in a manner that fully accounts for the complex dependencies that exist between weight settings and traffic flows.

1.1 Our Contributions

We propose a solution to the above problem that is fast, scalable and performs well without the drawbacks of previous techniques. Our contributions are three-fold. One, we introduce a new definition to classify the importance of a link failure that takes into account the traffic matrix, routing *and* the potential benefit to the network from protection against the failure. Two, we develop a new technique to quickly identify critical links based on this definition by extracting relevant information from standard weight search heuristics, like [5] and [4], used to optimize OSPF/IS-IS in no-failure scenarios. *Our technique uses the weight perturbations performed by these heuristics to approximate link failures and capture statistical identifiers, based on which it selects critical links*. Three, our technique possesses a novel feature in that it yields quantifiers for identification of links and/or networks whose failure performance is insensitive to weight settings. We show through computations that compared to [10], our technique yields a computational gain of a factor of over 10 for small size networks and a factor of 100-200 for large real-life sized networks. Equally important, we demonstrate that it yields better results that the techniques of [7] or [14] and is consistently faster.

The rest of the paper is organized as follows. Section 2 provides a general overview of the problem and the challenges involved, including a brief review of related works. Section 3 presents the heuristic we use for computing weight settings. Section 4 evaluates its performance and we conclude in Section 5.

2 Problem Formulation and Related Work

We model the network as a directed graph $G = (V, E)$ with $M = \|V\|$ vertices and $N = \|E\|$ edges. We assume the existence of a traffic matrix T which characterizes the traffic between each node-pair[4]. For a given set of link weights W, let $\Phi_0(W)$ denote the cost of routing the traffic matrix T over G, i.e., in the no-failure scenario, under IGP constraints. The choice of Φ_0 can be controlled by the network operator[5]. See Section 4 and [11] for details of the function used in this paper. Let $\Phi_l(W)$ denote the cost of routing T over $G-l$, i.e., when link l has failed [6]. Let F be some arbitrary function that weighs the importance of each no-failure and failure scenario. The IGP single link failure robust routing problem may then be stated as

$$\min_W F(\Phi_0, \Phi_{l_1}, \Phi_{l_2}, \ldots, \Phi_{l_N}) \quad ,W \in \mathbf{Z}^{+N} \tag{1}$$

For the purposes of this paper, we use

$$F(\Phi_0, \Phi_{l_1}, \Phi_{l_2}, \ldots, \Phi_{l_N}) = \Phi_0 + \mu \sum_{l \in E} \Phi_l \tag{2}$$

where μ is a parameter that weighs the importance of failures.

The equal-split constraint in standard IP forwarding renders the problem NP-hard [5] even in the (simpler) no-failure scenario and hence motivates the use of heuristics.

In the remainder of the section we review previous related work. The technique presented in [10] proposes a tabu-search heuristic that samples different weight settings and evaluates F over each sample, choosing the minimum. A major drawback of this method is that the computation of F requires evaluating *all* possible link failures for each weight setting. This quickly becomes intractable for large networks.

In order to overcome such computational bottlenecks, a common technique is to hypothesize that the total cost is dominated by a few variables and hence evaluate only their cost. A natural extension of this reasoning is to assume that only a *few* link failures are important and compute weight settings to protect only against their failures. This is equivalent to defining a new cost function

$$F_{E_C}(\Phi_0, \Phi_{l_1}, \Phi_{l_2}, \ldots, \Phi_{l_N}) = \Phi_0 + \mu \sum_{l \in E_C} \Phi_l \tag{3}$$

where the new set of links E_C satisfies $\|E_C\| \ll \|E\|$. If the hypothesis is indeed true, minimizing F_{E_C}, Eqn. (3) with a good choice of E_C should yield $F_{E_C}^{best} \approx F^{best}$ as well as significantly reduce computational complexity. However, this raises the question of how to carefully select the subset E_C so that performance is not compromised.

In [14], the links in E_C are determined via random selection. This ignores the relationship between the traffic T and the routing W. In [7] E_C is updated every few iterations by evaluating the cost Φ_l of *all* link failures and choosing the ones that incur the largest cost.

[4] The techniques developed in [6] for optimization over multiple traffic matrices are directly incorporable into our technique. For purposes of clarity and space, however, in this paper we focus on optimization for a *single* traffic matrix.

[5] It is typically a function of link load.

[6] We simply set $w_l = \infty$, re-compute any affected shortest paths and re-route the traffic matrix T.

This technique has two drawbacks. First, in some cases certain links may degrade network performance significantly on failure *regardless* of the weight setting. Including such links in E_C yields little reward in terms of computational effort since network performance would always be poor when these links fail. However, the definition used by [7] would likely include such links in the set E_C since their cost on failure is always large. The second drawback has to do with cases when the impact of a link failure *does* depend on weight settings. Since E_C is updated only every few weight changes, it is quite possible that E_C may not contain the critical links as defined by [7] with regards to the final chosen weight setting W^*. This can also lead to poor performance. We validate our observations in Section 4.

3 Heuristic for Failure Optimization

Based on our arguments in the previous section, we hypothesize that the criticality of a link, which decides its membership in E_C, should be a function of the range **and** variability of cost it incurs on failure over a large, if not the whole set of link weights. We now motivate and define a new metric for the *criticality* of a link that satisfies the above mentioned criteria and present a fast technique for its measurement.

Links which, regardless of weight settings *always* degrade network performance on failure, as well as links whose failure does not affect the network at all are considered *insensitive* to weight settings. Computational effort spent in finding weight settings to protect the network against failure of such links would yield very little reward. Hence such links should typically *not* be included in E_C. On the other hand, link failures that exhibit *large variations in cost across weight settings* would be ideal candidates for the critical link set (E_C). By carefully computing weights, one can protect the network against these link failures. Clearly their inclusion in the critical set yields increased returns in terms of performance improvement per unit of computational effort. These links are considered *sensitive* to weight settings. Hence, we define the *criticality* of a link as a function of some measure that is proportional to its *sensitivity* to weight settings.

In order to quantify the sensitivity of a link, we use a two-threshold based approach. We specify two thresholds to denote satisfactory and unsatisfactory performance. The sensitivity of a link is defined as the number of times, over all weight settings, that network cost on failure of the link falls below or above the satisfactory and unsatisfactory cost thresholds, respectively. The larger and close to each other the two numbers, the higher the likelihood that the link impacts network performance *and* responds to failure optimization. A simple example should illustrate this concept. Consider two links A and B, so that across 100 weight settings failure of link A results in a network cost that is below the satisfactory cost threshold (has little impact) 50 times and above the unsatisfactory cost threshold 50 times (large penalty). Conversely, for link B, the numbers are 90 and 10 respectively. Clearly, A is sensitive to weight settings, while the failure impact of link B is likely to be insensitive to weight settings. Hence inclusion of A in the critical set would be more beneficial.

It now remains to devise a technique that can determine the *sensitivity* of a link efficiently without having to compute the network cost after its failure for all possible weight settings. Two observations are key to our technique. First, a link failure may be approximated by an *increase* in link weight. Second, weight-search heuristics like [5] devised to compute optimal weights in no-failure scenarios typically operate by making numerous single weight changes to explore the weight space. We can exploit this to achieve our goals. During the course of the exploration of the weight space, we can approximate the impact of failure for each link over a large set of weight settings, simply by capturing appropriate statistical identifiers whenever a link weight is increased. This yields two advantages. One, we can compute the *sensitivity* of a link over a large set of routings *without* incurring additional complexity[7]. Second, it also provides an indicator on network robustness in general (see Section 3.4).

Once the *sensitivity* of each link has been computed, the most sensitive links are included in E_C. The same weight search heuristic is then re-run to compute weights that now minimizes the cost function F_{E_C}, Eqn. (3)

The heuristic can be thought of as consisting of 3 phases:

1. Phase 1: Run a standard weight search heuristic, e.g. [5], to compute the optimal link weights for the no-failure scenario (i.e. minimize Φ_0). During this search, collect statistics whenever link weights are large enough to simulate failure.
2. Phase 2: Compute *sensitivity* of each link and choose the most *sensitive* to comprise the set E_C.
3. Phase 3: Run the standard weight search heuristic to minimize the cost function, F_{E_C}, Eqn. (3).

We note that any heuristic that explores the OSPF/IS-IS weight space by perturbing link weights could be used in Phases 1 and 3. We used the algorithm presented in [5] as our template because the algorithm has been found to be quite fast and effective. In the next sub-sections, we briefly describe each phase. A detailed pseudo-code is available in [11] and omitted here due to space constraints.

3.1 Phase 1: Collecting Link Statistics

The main functions of Phase 1 are to get an estimate of the best cost in the no-failure case, Φ_0^{best} **and** to gather statistics to compute a measure of criticality for each link.

In each iteration, the underlying no-failure weight search heuristic begins with a weight setting W and then searches the neighbourhood of that space by choosing a link, and perturbing its link weight. Let the weight vector after perturbation be W'. Denote the cost after perturbation as $\Phi_0(W')$. Let the current estimate of the best no-failure cost is Φ_0^{best}. We denote the 2 thresholds mentioned in Section 3 to measure the sensitivity of a link by θ_1(unsatisfactory) and θ_2 (satisfactory).

Each time the perturbation results in the increase of weight of a link l, we collect the following statistics :

1. The number of times link weight *increases*. We denote this by $w_{inc}(l)$.

[7] Note that a weight search to optimize performance in the no-failure scenario needs to be run anyway since our objective contains Φ_0.

2. The number of instances for which the cost of the network lies above the threshold θ_1. We denote this estimate by $\Delta(l)$ and increment it by 1 whenever the current cost $\Phi_0(W')$ satisfies
$$\frac{\Phi_0(W') - \Phi_0^{best}}{\Phi_0^{best}} \geq \theta_1. \tag{4}$$
3. The number of instances for which the cost of the network lies below the threshold θ_2. We denote this estimate by $\gamma(l)$ and increment it by 1 whenever the current cost $\Phi_0(W')$ satisfies
$$\frac{\Phi_0(W') - \Phi_0^{best}}{\Phi_0^{best}} \leq \theta_2. \tag{5}$$

Observe that since $w_{inc}(l)$ is *always* incremented for a weight increase,

$$w_{inc}(l) \geq \Delta(l) + \gamma(l). \tag{6}$$

3.2 Phase 2: Selection of Critical Links

At the end of the execution of Phase 1, we have the estimate of the best no-failure cost Φ_0^{best}, as well as the necessary statistical identifiers to compute the sensitivity of links.

Phase 2 involves the creation of E_C. Towards this end we compute for each link l, the *frequency of cost change* $\alpha(l)$, which is our measure of sensitivity. This is done using the equation

$$\alpha(l) = \min\{(w_{inc}(l) - \Delta(l)), (w_{inc}(l) - \gamma(l))\} \tag{7}$$

The above equation can be explained as follows. If over all sampled weight settings, weight increases of a link result in cost increases that are either always above θ_1 or below θ_2, then the impact of this link's failure is relatively insensitive to weight settings. For such scenarios, if the cost of failure is always high (low), then $\Delta(l)$ (or $\gamma(l)$) would be large. From Eq. (6) and Eq. (7), $\alpha(l)$, the criticality of the link assumes a low value for both cases. On the other hand, if an increase in cost of the link results in widely varying cost increases that are often above or below the corresponding thresholds, then again from Eq. (6) both $\Delta(l)$ and $\gamma(l)$ will be comparatively small, resulting in a large value for $\alpha(l)$. Continuing with the example of Section 3, link A would have a criticality $\alpha(A) = \min(100-50, 100-50) = 50$, while link B would have a criticality $\alpha(B) = \min(100-90, 100-10) = 10$.

Construction of the critical link set E_C is now carried out as follows. All candidate links are sorted in decreasing order of the criticality $\alpha(l)$. The top L links, which are deemed to be the most critical links, are then chosen to constitute the set E_C of candidate links for failure optimization. $L = \| E_C \|$ is a parameter that can be specified.

3.3 Phase 3: Optimization over Critical Links

The final phase is responsible for finding robust weight settings to protect network performance against failures of links in the critical link set E_C. This is achieved simply by running the same weight search heuristic, but now to optimize the cost function F_{E_C} from Eqn. (3). The search is guided by rejecting all solutions whose no-failure cost,

Φ_0 exceeds the best no-failure cost Φ_0^{best} found in Phase 1 by more than $\theta_2\%$. This coupled with the fact that we are optimizing for failure of a small set of links reduces computational complexity.

3.4 Network Sensitivity to Failure

We now present an additional benefit of our method, namely, the statistical identifiers computed in Phase 1 can be used to estimate the robustness of a network as a *whole*. We define two *network* indicators

$$P = \frac{1}{\| E \|} \sum_{l: \frac{\Delta(l)}{w_{inc}(l)} \geq 0.95} \frac{\Delta(l)}{w_{inc}(l)}, \tag{8}$$

$$G = \frac{1}{\| E \|} \sum_{l: \frac{\gamma(l)}{w_{inc}(l)} \geq 0.95} \frac{\gamma(l)}{w_{inc}(l)}. \tag{9}$$

Intuitively, P (Pathological) provides an estimate of how many link failures can significantly degrade performance regardless of any weight setting, while G (Good) gives an estimate of the fraction of links that have a negligible impact on failure. One would expect G to be close to 1 in instances where almost no link failures cause performance to degrade. Alternatively, one would expect P to be close to 1, if the network is so heavily loaded, or the traffic poorly matched to the network, that almost every link failure causes significant degradation of performance. We shall see the relevance of those indicators in Section 4.

4 Experimental Evaluation

We evaluated the performance of our heuristic through computational experiments on various topologies and traffic matrices. The set of topologies used for evaluation include a PoP level version of the Sprint backbone as well as 5 synthetic networks generated using the Georgia Tech [15] topology generator. For the Sprint backbone, we used an actual traffic matrix generated from traffic measurements, while the traffic matrices for other networks were generated by picking intensities for each node-pair from a uniform distribution. Due to space constraints, we present only a subset of the results. Complete details may be found in [11].

A piecewise linear cost function introduced in [5] is used to represent the cost of routing traffic on each link. It is similar to an $M/M/1$ cost function in that it increases exponentially with link utilization. The network wide cost, Φ, is assumed to be the sum of all link costs. We refer the reader to [11] for further details on the cost function. The thresholds θ_1 and θ_2 were set to 100% and 20% of the best no-failure cost Φ_0^{best}, respectively. The weight search heuristic was run for 1000 iterations in Phase 1 and 100 iterations in Phase 3. As is standard practice in optimization, these parameters may be tuned further if required, although we found them to work well for all topologies considered.

For the smaller networks considered, we compare our algorithm (we shall refer to our heuristic as "Freq") against techniques presented in [10] (referred to as "Exhaustive"), [7] (referred to as "F&T"), random selection of E_C (referred to as "Random") as well as *optimal re-routing*. To obtain a fair comparison, the same weight search heuristic ([5]) used in our technique for exploring the weight space was also utilized in the other heuristics. For optimal re-routing, a new set of weights was computed for *all instances* of link failures on the reduced topology consisting of the original graph minus the failed link. We then compared the performance of our and other heuristics to the performance achievable when re-computing new routes after each failure.

For large networks, both the "Exhaustive" technique and computation of optimal routings for all of the $\| E \|$ possible failures are intractable. In order to circumvent this problem we first identify cases for which route re-computation has the potential to significantly improve performance, that is, those link failures that generate significant cost differences between the no-failure cost and that of our heuristic. For those and only those, we re-compute link weights (i.e recompute the routing) by running the no-failure weight search heuristic ([5]) on the truncated graph from which the failed link has been removed. This enables us to again compare the performance of the different heuristics (except the "Exhaustive") against a benchmark consisting of what is achievable when full re-routing is allowed.

Finally, we note that all the IGP weight search techniques ([5], [4] etc.) are random in nature and our identification of critical links is also based on statistical estimators. Hence it is possible that in a particular run we traverse an atypical sample path and make poor choices when selecting critical links, resulting in poor performance[8]. In order to quantify the frequency of such instances, we performed multiple runs (20) of all heuristics. Consequently, the results are presented in the form of the "fraction of times the heuristic cost lay within a certain fraction of the benchmark cost" over all the runs. The metric used for comparison was always the *worst case* deviation over all link failures between the heuristics and the benchmark costs. For example, a value of 60% under a column titled 30% would imply that in 60% of the runs, the *worst case* deviation of the heuristic from the benchmark over *all* link failures was less than 30%.

4.1 Small Networks

This sectionpresents results for a PoP level versionof theSprint network. Re-computation of the optimal routing after each failure was tractable and hence used as the benchmark. Table 1 shows results for a maximum link utilization of 0.7 under the optimal routing no-failure scenario. From the first row, we observe that the "Exhaustive search" heuristic performs quite well, always within 30% of *optimal re-routing*.

The next two rows, namely "Freq., $L = 40$" and "Freq. $L = 20$", show the performance of our heuristic with 40 and 20 critical links, respectively. As can be seen, the results are very good and comparable to those of the "Exhaustive Search" technique. 100% of the sample runs for $L = 40$, and 90% of the samples for $L = 20$, yield a performance within 40% of that obtainable with the best possible re-routing. The "Exhaustive Search" heuristic outperforms our heuristic with $L = 40$ in only 6.6% of the sample runs, and that too by less than 10%.

[8] Given their random nature, *all* the weight search techniques suffer from this problem.

Table 1. Comp. Results for various heuristics on the Sprint n/w, Max Util = 0.7

	Freq. of % Deviation from Optimal									
Heuristic	10%	20%	30%	40%	50%	60%	70%	80%	90%	$\geq$ 100%
Exhaustive	9.1	81.8	9.1	0	0	0	0	0	0	0
Freq. $L = 40$	13.33	73.33	6.67	6.67	0	0	0	0	0	0
Freq. $L = 20$	10	75	0	5	0	0	0	0	0	10
F&T $L = 40$	0	15	10	0	15	15	10	0	0	35
F&T $L = 20$	0	5	0	10	5	10	0	5	0	65
Random $L = 40$	0	25	20	0	20	10	0	0	0	25
Random $L = 20$	0	5	5	0	10	10	15	5	5	45
Plain	0	3.33	0	16.67	3.33	6.67	0	0	0	70

Table 2. Comp. Results for various heuristics on the Sprint n/w, Max Util = 0.5

	Frequency of % Deviation from Optimal									
Heuristic	10%	20%	30%	40%	50%	60%	70%	80%	90%	> 100%
Exhaustive	100	0	0	0	0	0	0	0	0	0
Freq. $L = 20$	100	0	0	0	0	0	0	0	0	0
F&T. $L = 20$	100	0	0	0	0	0	0	0	0	0
Plain	20	80	0	0	0	0	0	0	0	0

Our technique also outperforms the implemented “Random” as well as “F&T” techniques. From Table 1 we observe that the “F&T” technique has a worst-case deviation of more than 100% from optimal re-routing in 35% of the cases with $L = 40$ and in 65% of the cases with $L = 20$. “Random” selection of critical links results in a worst case deviation of more than 100% in 25% of the sample runs with $L = 40$ and in 45% of the sample runs with $L = 20$. This further re-enforces the importance of correctly selecting critical links. Finally, the last row (“Plain”), indicates performance when link weights are optimized only for the no-failure scenario. As expected, it performs the worst with 70% of the runs exceeding optimal performance by 100%.

Table 2 presents an interesting second sample point for the Sprint network, this time for a relatively low utilization of 0.5. As can be seen, all techniques including even the “Plain” heuristic performs exceedingly well. Why is this the case? Our network indicators P and G defined in Section 3.4 correctly explain this situation. The P and G indicators for the various networks at various levels of utilization have been computed in Table 3(a). Note from the 2nd entry, which represents the network in question, that it has a G entry of 0.98. This indicates that at the current utilization, the network should be relatively insensitive to failures, which is indeed the case. Note also that the 1st row which represents the Sprint network with the previous utilization of 0.7, has a P value of 0.046. This indicates the presence of some (4) links that always cause degradation on failure. We confirmed this observation for all 4 links by comparison with the optimal performance on failure of these links.

Our technique is the only one that allows the explicit identification of such instances without resorting to expensive computation. The P and G network indicators readily

Table 3. Computational Time and Insensitivity Indicators for Sprint n/w

Row	Network	Max.	Insensitivity	
		Util.	P	G
1.	Sprint	0.7	0.046	0
2.	Sprint	0.5	0	0.98
3.	150 Node	0.7	0	0.97

(a) Insensitivity of various network-traffic matrix instances to link failure

Heuristic	Avg. Time(secs)	Max. Time(secs)
Exhaustive	3320	4207
Freq. $L = 40$	510	626
Freq. $L = 20$	189	224
F&T $L = 40$	608	743
F&T $L = 20$	270	546
Plain	36	49

(b) Comp. Times for various heuristics

identify instances where failure optimization is not beneficial, simply by using information from the computationally inexpensive Phase 1. Based on the values of P and G, one can decide to skip the more expensive Phase 2, and/or initiate a redesign of the network.

Table 3(b), displays statistics of the computation time (on a Dell Intel 1.5 GHz) for each heuristic on the Sprint network. The table clearly demonstrates the significant gain in computation time of our heuristic over the “Exhaustive Search” technique. Our heuristic takes an order of magnitude less time to compute solutions, and in most cases yields equally good weight settings. We also observe that the “Freq.” heuristic is faster than the “F&T” technique.

4.2 Large Networks

Scalability to cope with the large size of present day networks has been an important goal of our heuristic. By virtue of the size of large networks, one expects a link failure to have less impact than on small networks. Hence we use smaller critical link sets to improve the running time. Table 4(a) shows results for a 150 node 432 edge graph for “Freq.” and “F&T” heuristics with $L = 10$, and for the “Plain” heuristic. Our heuristic as well as the “F&T” heuristic perform well, within 10% of the benchmark over all runs, while the “Plain” heuristic performs well in 80% of its runs. The results are attributable to the high value of $G = 0.97$ for the 150 node network. All but 4 of the links cause little impact on failure. However the 4 links are sensitive to weight settings which offsets the “Plain” heuristics performance in 20% of the runs.

Running times for the network are shown in Table 4(b). Our heuristic took about 4.6 hours and was much faster than the “F&T” heuristic. For the network in question, 10 iterations of the “Exhaustive Search” took about 7 hours. The underlying weight search scheme ([5]) typically takes about 1000 iterations to produce consistent results ([5]). A straightforward extrapolation indicates that the “Exhaustive” technique would require 700 hours to produce results equivalent to our method, i.e., about 2 orders of magnitude more.

Results for several other small and large networks are presented in [11]. They further demonstrate the speed and effectiveness of our technique over other methods as well as the benefits of the network indicators P and G.

Table 4. Performance and complexity statistics for 150 node n/w

	Freq. of % Dev. from No-Failure Cost		
Heuristic	10%	20%	> 100%
Freq. $L = 10$	100	0	0
F&T $L = 10$	100	0	0
Plain	0	80	20

(a) Comp. Results for various heuristics, Max Util = 0.7

Heuristic	Avg. Time(secs)	Max. Time(secs)
Freq. $L = 10$	16489	17847
F&T. $L = 10$	23800	138646
Plain	5559	6387

(b) Comp. Times for various heuristics

5 Conclusions

We have addressed the problem of computing *a single set* of link weights for IGP routing to protect the network against single link failures and perform well under normal conditions. This is an important problem because single link failures are the most common type of failures, and can generate significant disruption if not properly guarded against. The use of "proactive" solutions limits the amount of overhead and re-routing required and also minimizes disruption which are desirable given the transient nature of such failures.

In this work we have presented an efficient and novel heuristic that exploits the information gathered by no-failure search heuristics. We showed how to use this information to construct a set of critical links that are most likely to degrade performance on failure. Exhaustive search to optimize for failure is then performed only for this smaller set of links. Through computational evaluations we have demonstrated that the heuristic scales well with network size, and outperforms previous approaches. Finally, a novel feature of our heuristic is in the form of the P and G indicators that provide early indication of the general sensitivity of the network to failures.

References

1. B. Albrightson, J.J. Garcia-Luna-Aceves, and J. Boyle. EIGRP - a fast routing protocol based on distance vectors. In *Proceedings of Networld/Interop*, May 1994.
2. W. Ben-Ameur. Multi-hour design of survivable classical IP networks. *International Journal of Communication Systens*, 15:553–572, 2002.
3. R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. Request For Comments (Standard) RFC 1195, Internet Engineering Task Force, December 1990.
4. M. Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. In *Journal of Combinatorial Optimization*, volume 6, 2002.
5. B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of INFOCOM'2000*, Tel Aviv, Israel, March 2000.
6. B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communication*, May 2002.
7. B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In *W. Ben-Ameur and A. Petrowski, Editors, Proceedings of INOC*, pages 225–230, October 2003.

8. G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharya, and C. Diot. Analysis of link failures in an IP backbone. In *Proceedings of IMW 2002*, Marseilles, France, November 2002.
9. J. Moy. OSPF Version 2. Request For Comments (Standard) RFC 2328, Internet Engineering Task Force, April 1998.
10. A. Nucci, B. Schroeder, S. Bhattacharya, C. Diot, and N. Taft. IGP link weight assignment for transient link failures. *Proceedings of the 18th ITC*, August 2003.
11. A. Sridharan and R. Guérin. Making OSPF/IS-IS Routing Robust to Link Failures. Technical report, Available at http://einstein.seas.upenn.edu/mnlab/publications.html, University of Pennsylvania, July 2004.
12. A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. *Proceedings of INFOCOM'2003*, April 2003.
13. Z. Wang, Y. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proceedings of INFOCOM'2001*, Anchorage, Alaska, April 2001.
14. Di Yuan. A Bi-Criteria Optimization Approach for Robust OSPF Routing. Technical report, Linkoping University, 2003.
15. E. W. Zegura. GT-ITM: Georgia Tech Internetwork Topology Models (software). http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm, Georgia Tech, 1996.

Interdomain Ingress Traffic Engineering Through Optimized AS-Path Prepending

Ruomei Gao, Constantinos Dovrolis, and Ellen W. Zegura

College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332*
{gaorm, dovrolis, ewz}@cc.gatech.edu

Abstract. In INterdomain Ingress Traffic Engineering (INITE), a "target" Autonomous System (AS) aims to control the ingress link at which the traffic of one or more upstream source networks enters that AS. In practice, ISPs often manipulate, mostly in a trial-and-error manner, the length of the AS-Path attribute of upstream routes through a simple technique known as prepending (or padding). In this paper, we focus on prepending and propose a polynomial-time algorithm (referred to as OPV) that determines the optimal padding for an advertised route at each ingress link of the target network. Specifically, given a set of "elephant" source networks and some maximum load constraints on the ingress links of the target AS, OPV determines the minimum padding at each ingress link so that the load constraints are met, when it is feasible to do so. OPV requires as input an AS-Path length estimate from each source network to each ingress link. We describe how to estimate this matrix, leveraging the BGP Looking Glass Servers. To deal with unavoidable inaccuracies in the AS-Path length estimates, and also to compensate for the generally unknown BGP tie-breaking process in upstream networks, we also develop a robust variation (RPV) of the OPV algorithm.

1 Introduction

Traffic Engineering (TE) is broadly divided into two types: intradomain and interdomain. In intradomain TE, the operator of an autonomous system (AS) controls the flow of traffic within that network by optimizing the link costs of the corresponding routing protocol (mostly OSPF or IS-IS) or through dynamic provisioning of virtual circuits (e.g., MPLS). For previous work in intradomain TE, we refer the reader to the survey [1] and the references therein. Intradomain TE assumes that the ingress and egress links of interdomain traffic flows are given as inputs in the form of a traffic matrix, and they cannot be manipulated.

Interdomain TE, on the other hand, aims to control exactly those ingress and egress flows. Let us consider a "target" AS, referred to as $\mathcal{T}$. $\mathcal{T}$ has a number of ingress links, receiving traffic from upstream ASes. If $\mathcal{T}$ is not a stub network, traffic that is destined to one of that network's customers eventually leaves $\mathcal{T}$ through an egress link. Controlling the egress link that the traffic will flow through is referred to as *Interdomain Egress*

* This work was supported by the NSF CAREER award ANIR-0347374 and by an equipment donation from Intel Corporation.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 647–658, 2005.

TE. On the other hand, controlling the ingress link through which the traffic of a source network will enter $\mathcal{T}$ is referred to as *INterdomain Ingress TE (INITE)*.

If one compares the maturity, in terms of both operations and research, between INITE and other types of TE, the former is much less deployed, understood, and trusted [2, 3]. The high-level reason is that INITE requires that the target network $\mathcal{T}$ has a way to influence BGP routing decisions in upstream ASes, without requiring the active cooperation of those networks. In more detail, to perform INITE the operator of $\mathcal{T}$ would face the following three major unknowns about the *upstream cloud*, i.e., the part of the Internet between a source network and the ingress links of $\mathcal{T}$. First, the *BGP policies*, expressed through BGP attributes (e.g., Local Preference) and upstream ingress/egress routing filters. Second, the actual AS-level topology and in particular, the *AS-Path lengths* from any upstream network to the ingress links of $\mathcal{T}$. Third, the *BGP tie-breaking behavior*, referring to the way a BGP peer selects the best route among a set of candidate routes with the same Local Preference and AS-Path length [4]. In addition to the previous unknowns, one has to include the more common challenges of any TE problem, including variations in the traffic loads, unexpected infrastructure events (e.g., router crashes), and so forth.

Given the previous difficulties, it is not surprising that some ISPs avoid INITE. One form of INITE that is, however, used by many ISPs is that of *AS-Path prepending*, or simply "prepending" (also known as *AS-Path padding*). Specifically, $\mathcal{T}$ can make a route less attractive to its upstream ASes by increasing the AS-Path length of that route by adding several instances of its own AS-Number to that attribute. A recent measurement study showed that 32% of the routes in the AT&T network have some form of prepending, with about 90% of the corresponding paths extended by 1-5 hops [3]. Prepending is often used for load balancing and for provisioning of backup routes. In the former, operators increase the degree of padding at an advertised route in a trial-and-error basis until the AS-Path is long enough to reduce the load of that ingress link by a certain amount. In the latter, the degree of padding is sufficiently large so that the route is not used unless if there is a failure in the primary route(s).

Our objective is to investigate the prepending technique more thoroughly, and understand its potential and limitations. The specific problem that we consider, in its basic form, is the following. Suppose that a target network $\mathcal{T}$ has N ingress links. For simplicity, we will assume that $\mathcal{T}$ is a multihomed stub network. The case in which $\mathcal{T}$ is a transit network, performing INITE for its customers, is considered in [5]. $\mathcal{T}$ receives most of its traffic from a set of M "elephant" sources, and an estimate of each source load s_i is given (i=1 $\ldots M$). $\mathcal{T}$ aims to impose a *maximum load* (maxload) constraint C_j at each ingress link j for the traffic that arrives from the M sources. To do so, $\mathcal{T}$ increases the AS-Path length of its advertised routes at ingress link j by prepending its own AS-Number $a_j \geq 0$ times. The main questions then are: *what is the value of a_j for each link j that will meet the given maxload constraints, and when is it infeasible to meet these constraints through prepending?* In particular, we are interested in the *optimal prepending*, i.e., the padding vector A that minimizes the sum $\sum a_j$. We refer to the previous as the *Constrained Optimal Prepending (COP)* problem. COP, despite its simple statement, captures an important objective of multihomed ASes, that of balancing the ingress load across a set of links, and it attempts to leverage a currently ad-hoc technique (prepending) in a more systematic methodology.

The first contribution of this paper is to develop a polynomial-time algorithm, referred to as *OPV* (for Optimal Padding Vector), which solves the COP problem. The algorithm is very simple: at each iteration, an overloaded ingress link is chosen and its padding is incremented. Then, given the new padding vector, the mapping from sources to ingress links is recomputed, and the algorithm moves to the next iteration. A key input to the OPV algorithm is the *AS-Path length matrix* P. Each element $P_{i,j}$ of this matrix is an estimate of the shortest AS-Path length from source network $\mathcal{S}_i$ to ingress link $\mathcal{L}_j$. The second contribution of the paper is to present four estimation techniques for P, leveraging the abundant Looking Glass Servers present in the Internet today, and to evaluate the accuracy of these techniques.

The errors of the AS-Path length estimation are not negligible however. To deal with errors in P, our third contribution is to develop the *Robust Padding Vector (RPV)* algorithm, a heuristic built on top of OPV. The objective of RPV is to determine a padding vector that will likely satisfy the given maxload constraints, even if the input matrix P has inaccurate elements and the BGP tie-breaking behavior is unknown in the upstream cloud. Simulation results show that, with the observed empirical error distribution in P, and the completely unknown tie-breaking behavior, RPV can still find a padding vector that satisfies the maxload constraints in more than 90% of the cases, as long as such a padding vector exists.

Related Work. A survey for interdomain TE is [2]. The feasibility of interdomain TE by a stub AS was examined in [6] and more recently in [7]. A simulation-based study of the effectiveness of AS-Path prepending for INITE has been published in [8]. The use of the BGP community attribute for INITE has been studied in [9]. The Internet Draft [10] proposed another BGP community-based solution. More recently, [11] proposed a tunneling-based mechanism for INITE, assuming the cooperation of the remote source networks. Instead of adding extensions to BGP, Agarwal et al. proposed an Overlay Policy Control Architecture (OPCA) [12]. Feamster et al. focused on egress interdomain TE [3]. Uhlig et al. also focused on egress interdomain TE in [13]. Bressoud and Rastogi examined the intradomain problem of choosing the optimal set of border routers for the advertisement of a set of routes from a transit provider [14].

Paper Structure. The COP problem is formally stated in §2, In §3, we propose and study the OPV algorithm, proving that it finds the optimal padding vector in polynomial time. In §4, we describe how to estimate the AS-Path length matrix and evaluate the accuracy of the proposed techniques. In §5, we present the RPV algorithm and evaluate its robustness through simulations. We conclude in §6.

2 Problem Statement and Formulation

Consider a target network $\mathcal{T}$ that aims to do INITE through prepending. INITE can be performed separately for each major customer of $\mathcal{T}$, if the latter is a transit network. Or, if $\mathcal{T}$ is a multihomed stub network, INITE can be performed for all the ingress traffic to $\mathcal{T}$. Suppose that $\mathcal{T}$ has N ingress links $\{\mathcal{L}_1, \ldots, \mathcal{L}_\mathcal{N}\}$, each of which advertises the

Table 1. Notation

$\mathcal{T}$	target network	M	number of sources
$\mathcal{S}$	source network vector	N	number of ingress links
$\mathcal{L}$	ingress link vector	$\mathcal{U}_i$	upstream tree for src i
S	source load vector	C	link maxload vector
P	AS-Path length matrix	Q	tie-breaking matrix
A	padding vector	$P(A)$	padded length matrix
$L(A)$	link assignment vector	$R(A)$	link load vector

Table 2. BGP route selection

1. Higher local preference
2. Shorter AS path
3. Lower origin type
4. Lower MED value
5. E-BGP over I-BGP routes
6. Lower IGP metric to next-hop
7. Lower BGP router ID

routes that originate at $\mathcal{T}$ to its upstream network through BGP. The ingress traffic for $\mathcal{T}$ may be produced by a potentially large number of upstream source networks. Instead of considering individual source networks, that may be impractical, we focus on *super-source ASes*. The latter may be an aggregation of several source networks that generate a relatively large portion of the total ingress traffic to $\mathcal{T}$. In the following, we refer to super-source ASes simply as "source networks". Suppose $\mathcal{T}$ has M source networks, and the average traffic loads from the M source networks are represented by the *source load vector* $S = \{s_1, \ldots s_M\}$.

The effectiveness of any AS-Path prepending technique, including ours, is limited by the use of local routing policies expressed through the Local Preference attribute. The reason is that the Local Preference attribute has a higher priority in the BGP path selection process than the AS-Path length (see Table 2). Consider a source network i, and suppose that i maintains a distinct route to each ingress link $\mathcal{L}_j$ of $\mathcal{T}$. This can be achieved if $\mathcal{T}$ advertises a unique *wayfinding prefix* at each link $\mathcal{L}_j$ (we return to this point in § 4). The selected routes from source i to the N ingress links of $\mathcal{T}$ form a directed acyclic graph (DAG) of AS links denoted by $\mathcal{U}_i$. In the following, we assume that the branching nodes of this DAG select their best routes to the N ingress links only based on the AS-Path length. In other words, the candidate routes for $\mathcal{T}$ at each branching node of $\mathcal{U}_i$ are assigned the same Local Preference value.

The BGP tie-breaking behavior is far from simple. As shown in Table 2, when two routes have the same Local Preference and AS-Path length, an ordered list of five other criteria is used to choose the best route. We do not attempt to estimate or infer all the parameters that affect those tie-breaking criteria; such a task would be extremely difficult and error prone. We assume, however, that the tie-breaking behavior is determined by a matrix Q. The i'th row of Q refers to the tree that connects the i'th source network to the N ingress links of $\mathcal{T}$. If two routes that originated from links $\mathcal{L}_j$ and $\mathcal{L}_k$ are received with equal AS-Path lengths by an AS in the upstream tree $\mathcal{U}_i$, then that AS will choose the route to link $\mathcal{L}_j$ if $Q_{i,j} < Q_{i,k}$; otherwise, it will choose the route to link $\mathcal{L}_k$. Different columns of the same row of Q must be different, while their absolute values do not matter. Notice that Q is just a model of the BGP tie-breaking behavior; it is not related to a real BGP attribute. Furthermore, Q represents a *globally consistent tie-breaking behavior*, i.e., we assume that a tie between two routes to $\mathcal{T}$ with the same AS-Path length is broken in the same way in any AS in an upstream tree $\mathcal{U}_i$.

Another key parameter is the *AS-Path length matrix* P. The element $P_{i,j}$ is the length of the shortest AS-Path from the source network $\mathcal{S}_i$ to the ingress link $\mathcal{L}_j$, where

$P_{i,j}$ is a positive integer. Any ties in the length of the shortest AS-Path are broken based on the Q matrix. Note that $P_{i,j}$ is the "original" AS-Path length, i.e., it should be measured before the target network applies any prepending. Estimation techniques for the matrix P are given in § 4. For now we assume that P is accurately known.

The ingress link $\mathcal{L}_j$ can increase the length of the AS-Path attribute by a_j, where a_j is a non-negative integer, through prepending. Given a padding vector A=$\{a_j, j = 1 \ldots N\}$, the "padded" AS-Path length of the route to link $\mathcal{L}_j$ is $P_{i,j}(A) = P_{i,j} + a_j$[1].

Based on the previous model, we can now show the following.

Lemma 1. *Given a padded AS-Path length matrix $P(A)$ and a tie-breaking matrix Q, the traffic of a source network $\mathcal{S}_i$ will enter the target network $\mathcal{T}$ through the ingress link $\mathcal{L}_j$, where*

$$j = \arg \min_{1 \leqslant l \leqslant N} P_{i,l}(A) \tag{1}$$

If there is a tie between links $\mathcal{L}_j$ and $\mathcal{L}_k$, then $Q_{i,j} < Q_{i,k}$.

Due to space constraints, the proofs are given in the extended version of this paper [5].

Based on the previous Lemma, we can now define the *link assignment vector* $L(A)$= $\{l_i(A), i = 1 \ldots M\}$, where $l_i(A)$ is the link $\mathcal{L}_j$ that source $\mathcal{S}_i$ selects based on Lemma 1. From the link assignment vector $L(A)$, the expected load at an ingress link $\mathcal{L}_j$ is

$$r_j(A) = \sum_{k=1 \ldots M : l_k(A) = \mathcal{L}_j} s_k \tag{2}$$

The vector $R(A)$=$\{r_j(A), j = 1 \ldots N\}$ is the *link load vector*. The type of INITE that we consider is based on a set of maximum load constraints for each ingress link. Specifically, let c_j be the maximum traffic load (maxload) allowed at link $\mathcal{L}_j$, with C=$\{c_j, j = 1 \ldots N\}$ being the corresponding *link maxload vector*. A link $\mathcal{L}_j$ is *overloaded* if $r_j(A) > c_j$. When none of the N ingress links is overloaded, we say that A is *acceptable* under C.

The COP problem can now be stated as follows. *Given an instance I=(S, C, P, Q), determined by the source load vector S, the link maxload vector C, the AS-Path length matrix P, and the tie-breaking matrix Q, does an acceptable padding vector A exist? When it is true, determine the* optimal padding vector A^**, such that across all acceptable padding vectors A* $\sum_{j=1}^{N} a_j^* \leq \sum_{j=1}^{N} a_j$. When there is an acceptable padding vector for a given instance I, we say that I is *feasible*; otherwise, I is *infeasible*.

3 Optimal Padding Vector Algorithm

The OPV algorithm (Algorithm 1) takes as input an instance I=(S, C, P, Q) of the COP problem, and examines the feasibility of a single padding vector $A^{(m)}$ in iteration m, starting from the zero padding vector. With each padding vector that is not acceptable, the algorithm identifies an overloaded link and then increments the corresponding

[1] Some ISPs have an agreement with their peers that they will announce the same AS-Path to a certain destination through all their peering links. If that is the case, $\mathcal{T}$ can group together all ingress links to each peer, and then apply the proposed algorithms at those link groups.

Algorithm 1.OPV (I=(S, C, P, Q))

```
Compute $\Phi$ from (3) and (4)
$A^{(0)} = [0, \ldots, 0]$;
for $m = 0$ to $\Phi$-1 do
    Compute $L(A^{(m)})$ from (1)
    Compute $R(A^{(m)})$ from (2)
    if $A^{(m)}$: acceptable (i.e., $\forall j, r_j(A^{(m)}) \leqslant c_j$) then
        return $A^{(m)}$
    end if
    $A^{(m+1)} = A^{(m)}$
    Identify an overloaded link $j$, i.e., $r_j(A^m) > c_j$
    $a_j^{(m+1)} = a_j^{(m)} + 1$
end for
return $I$: Infeasible
```

padding element. The exact selection of the overloaded link does not matter. The algorithm exits either when it has found an acceptable padding vector, or when it has completed Φ iterations, the upper bound of the number of iterations.

The bound Φ, which is proven to be a tight bound in Lemma 3 and Lemma 4, is computed as follows:

$$\phi_j = \max_{1 \leqslant s \leqslant M} \max_{1 \leqslant i \leqslant N} (P_{s,i} - P_{s,j}) \tag{3}$$

and

$$\Phi = N + \sum_{1 \leqslant j \leqslant N} \phi_j - \min_{1 \leqslant j \leqslant N} \phi_j \tag{4}$$

Note that Φ can be computed in polynomial time.

Later in this section, we show two important properties of OPV. First, when I is feasible, OPV reports the optimal padding vector A^*. Second, when I is infeasible, OPV exits after Φ iterations reporting that indeed, there is no acceptable padding vector.

The following lemmas prove that for a feasible instance: first, the optimal padding vector A^* has at least one zero element; second, when a padding element a_k reaches the value of the corresponding element in the optimal padding vector at iteration m, i.e., $a_k^{(m)} = a_k^*$, link k will not be overloaded in any subsequent iterations, and so a_k remains at a_k^*; third, each optimal padding element is upper bounded by a function of P that can be computed in polynomial time.

Lemma 2. *Suppose that the instance I is feasible, with optimal padding vector A^*. There exists a link j with $a_j^* = 0$.*

Lemma 3. *Suppose that the instance I is feasible, with an optimal padding vector A^*. If the OPV padding vector in iteration m is such that $a_j^{(m)} \leqslant a_j^*$ for all links j and there exists a link k such that $a_k^{(m)} = a_k^*$, then in all subsequent iterations n ($n > m$), $a_k^{(n)} = a_k^{(m)} = a_k^*$.*

Lemma 4. *Suppose that the instance I is feasible, with optimal padding vector A^*. Then, for any link j, the corresponding optimal padding is bounded by $a_j^* \leqslant \phi_j + 1$, where ϕ_j is given by (3).*

The following theorems give the main result for the OPV algorithm in the case of feasible instances and infeasible instances, respectively.

Theorem 1. *Suppose that the instance I is feasible, with optimal padding vector A^*. The OPV algorithm returns the vector A^* as its final outcome in polynomial time.*

Theorem 2. *If instance I is infeasible, then the OPV algorithm detects that there is no acceptable padding vector in polynomial time.*

4 Estimation of Input Parameters

In this section, we focus on the two key unknown inputs of the OPV algorithm, namely the set of super-source networks and the corresponding source load vector S, and the length estimation matrix P. As mentioned in §2, we do not attempt to estimate the tie-breaking matrix Q; the algorithm of the following section determines a robust padding vector independent of Q. Also, the maxload vector C is supposed to be known, given that it is chosen by the target network operator.

4.1 Selection of Super-Sources

We assume that the ingress routers of $\mathcal{T}$ collect statistics (with Cisco's NetFlow for instance) of the arriving traffic, aggregated by source networks. Since the ingress traffic may originate from many source networks, keeping track of the traffic by individual source networks may not be feasible at $\mathcal{T}$. We reduce the number of sources that need to be monitored by first aggregating several source networks into a super-source, and then only collecting volume statistics for large super-sources. The aggregation of source networks is based on the knowledge of the AS-level topology, which can be constructed with multiple vantage points [15]. For example, NetFlow data may show that several major sources of traffic for $\mathcal{T}$ belong to three stub ASes that are all single-homed customers of a tier-2 provider AS-X. In that case, AS-X can be considered as a super-source $\mathcal{S}_i$. The corresponding source load element s_i would be estimated as the sum of the load estimates of the actual sources, measured with NetFlow. To deal with the variability and unpredictability in the traffic volume of each source network, the capacity of an ingress link can be increased so that it has a sufficient headroom.

4.2 Estimation of AS-Path Length Matrix

To estimate $P_{i,j}$, $\mathcal{T}$ has to originate a route for a *wayfinding prefix* at each ingress link $\mathcal{L}_j$. A wayfinding prefix can be just the IP address of $\mathcal{L}_j$. If that is an unacceptably long prefix, the wayfinding prefix can cover instead a small part of the target network's address space. To estimate $P_{i,j}$, the operator of $\mathcal{T}$ can use one of the four following techniques, in the given order (see Figure 1). The first three techniques rely on *Looking*

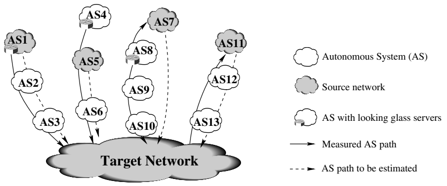


Fig. 1. Four estimation techniques for the AS-Path length

Glass Servers (LGS). LGS's are abundant in the Internet today, providing a "peek" inside a network's BGP routing tables. They are mostly used for problem diagnosis and monitoring of interdomain routing. The publicly available LGS's that are listed in *www.traceroute.org* include 273 servers that in some cases provide access to multiple routers within an AS. It is likely however that major ISPs have private access to even more LGS's in remote networks, to accommodate synergistic problem diagnosis.

- **LGS in $\mathcal{S}$:** The source network $\mathcal{S}$ may include an LGS. This ideal case leads to the most accurate estimation. In the example of Figure 1, AS1 is a source network that deploys an LGS. $\mathcal{T}$ can just query that LGS for the AS-Path to wayfinding prefixes.
- **LGS in a customer of $\mathcal{S}$:** Here, a customer of $\mathcal{S}$ deploys an LGS. In that case, $\mathcal{T}$ can query that LGS for each wayfinding prefix, and then remove from the returned AS-Paths the part that includes that customer AS. In Figure 1, AS4 is a customer of the source network AS5. AS4 deploys an LGS.
- **LGS in the path from $\mathcal{T}$ to $\mathcal{S}$:** Another possibility is that $\mathcal{T}$ can locate an LGS in a network in the reverse path, from $\mathcal{T}$ to $\mathcal{S}$. In Figure 1, AS7 is a source network and AS8 is a network deploying an LGS in the reverse path. In that case, $\mathcal{T}$ can estimate the AS-Path length $P_{i,j}$ as the sum of the AS-Path lengths from the LGS to $\mathcal{L}_j$ and from the LGS to $\mathcal{S}$. This technique assumes that the AS-Paths from the LGS to $\mathcal{S}$ and from $\mathcal{S}$ to the LGS have the same length.
- **Reverse path estimation:** When none of the previous techniques is applicable, $\mathcal{T}$ can just estimate the length of the AS-Path from $\mathcal{S}$ to $\mathcal{L}_j$ based on the length of the reverse path from $\mathcal{L}_j$ to $\mathcal{S}$, which is of course directly available at the border router at $\mathcal{L}_j$. The problem with this technique is that it relies on the symmetry of the forward and reverse AS-Paths, which is often not the case in the Internet. On the positive side, the technique still produces a correct estimate if the two paths have the same length, even if those paths are different.

Note that we cannot use the AS-level traceroute tool [16, 17] in the estimation of the P matrix, because that technique cannot report the degree of padding in an AS path.

AS-Path length distribution and estimation errors: The first two of the four previous techniques would give no estimation errors, because the corresponding LGS's report directly the AS-Path from the source to the ingress links of the target network. The third and fourth techniques, on the other hand, depend on the symmetry assumption, and they can suffer from estimation errors. To quantify these errors, we set up an experiment using 79 of the publicly available LGS's listed at *www.traceroute.org*. In this experiment, we estimate the AS-Path length from every LGS to each of the 78 other LGS's using the previous two techniques: "LGS in the reverse path" (when applicable), and "reverse path estimation" (always applicable). Then, we compare each AS-Path length estimate with the length of the actual AS-Path, as that is reported in the remote LGS.

The unconditional estimation error for the previous two estimation techniques is as follows: 60-65% of the estimates have zero error, 85-90% have an error of less than ± 1 hop, and 95% have an error of less than ± 2 hops. More importantly, we calculate the conditional probability of an estimation error e, given the estimated AS-Path length $\hat{p}$. The estimation error e is measured as $\hat{p} - p$, where p is the actual AS-Path length. The conditional probability that an estimate is error-free if $\hat{p}$ is 3 or 4 hops is 78% and 70%, respectively, which is higher than the corresponding unconditional accuracy. Similarly, the conditional probability that an estimate has an error of less than ± 1 hop if $\hat{p}$ is 5 or 6 hops is 88% and 86%, respectively. We also use the LGS servers to measure the AS path length distribution. About 90-95% of the paths are up to 6 AS hops, including any prepending, while 60% of the paths are either 4 or 5 hops. We use this AS-Path length distribution, as well as the previous estimation error conditional probabilities, in the robustness study of the next section.

5 Robust Padding Vector Algorithm

In this section, we first describe an algorithm that aims to determine a robust padding vector in the presence of these errors, and then evaluate the effectiveness of that algorithm with simulations.

5.1 Robust Padding Vector Algorithm

Recall that an instance I of the COP problem is defined by the set (S, C, P, Q). The OPT algorithm of §3 was developed to find an optimal padding vector A^* for I. In practice, we do not have a way to estimate Q, and P may include estimation errors. To deal with these two issues, we develop the *RPV (Robust Padding Vector)* algorithm. The basic idea in RPV is the following. Suppose that P_0, our original estimate of P, together with an arbitrary tie-breaking matrix Q and the given S and C vectors, form the instance I_0 that we start from. The unknown *actual instance* I_a, based on the correct AS-Path length matrix and the real tie-breaking behavior[2], belongs in the space $\mathcal{I}$ of all possible instances that can be generated by applying a given error model to P_0, and by considering all possible tie-breaking behaviors.

First, RPV generates a subset $\mathcal{I}_X$ of $\mathcal{I}$ that consists of X feasible instances. For each instance I_i in $\mathcal{I}_X$, the corresponding optimal padding vector (computed with OPV) is A_i. The set of padding vectors $\mathcal{A}$=$\{A_i, i = 1 \ldots X\}$ is our candidates for the desired robust solution. To generate padding vectors that differ significantly, we create the subset $\mathcal{I}_X$ by applying the AS-Path length estimation error model only to the largest sources of I_0. Errors that correspond to small sources (relative to the rest of the sources) may not have an impact on the resulting padding vector. Second, RPV generates a large

[2] We still assume that the real tie-breaking behavior can be captured by a matrix such as Q.

subset $\mathcal{I}_Y$ of $\mathcal{I}$ that includes Y instances. The fraction of feasible instances in $\mathcal{I}_Y$ is the *feasibility index* of $\mathcal{I}_Y$. If Y is sufficiently large, the feasibility index estimates the probability that the actual instance I_a is feasible. Third, for each candidate padding vector A_i in $\mathcal{A}$, RPV measures the fraction of *feasible* instances in $\mathcal{I}_Y$ for which A_i is acceptable. That fraction is the *robustness index* of A_i. If Y is large, the robustness index estimates the probability that the corresponding padding vector A_i is acceptable for the actual instance I_a, conditioned on the fact that the latter is feasible.

Eventually, RPV reports the padding vector $\tilde{A}$ with the maximum robustness index. The reason is that that vector maximizes the likelihood that it will be acceptable for I_a. The higher Y is, the more samples we collect from the space of possible instances, and so the more reliable our robustness index will be. The robustness, on the other hand, can be increased if we increase X.

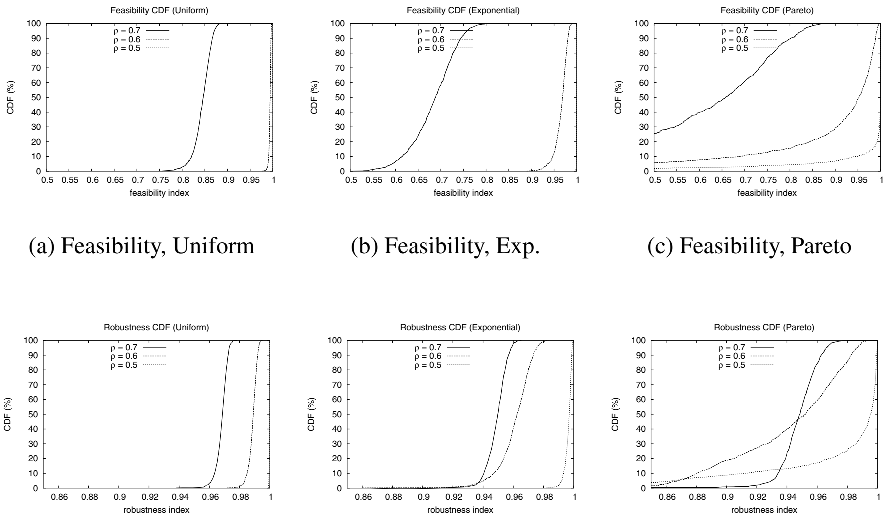
5.2 Robustness Evaluation

We evaluate the robustness of the padding vector that RPV reports using simulations. In the following, we consider a target network with N=5 ingress links that have the same maxload constraint (i.e., C_j=C for all j). The number of source networks is M=100. The source load distribution is the same for all sources, and it follows one of the following models: Uniform, Exponential, and Pareto (shape parameter: 1.7). The mean source load $\bar{s}$ is the same in all distributions. The AS-Path length matrix P_0 is generated based on the LGS empirical distribution mentioned in §4. The error model applied to P_0 is based on the conditional estimation errors, which are collected through the measurement described in §4, with given AS-Path lengths. The RPV algorithm uses X=100 and Y=10000 instances. $\mathcal{I}_X$ is formed by applying errors to the set of sources that generate, in total, at least 80% of the aggregate load.

The feasibility index depends on the relation between the aggregate offered load $M\bar{s}$ and the aggregate maxload constraint NC. Given the source load vector S, we define our *load metric*, $\rho = \frac{\sum_{i=1}^{M} s_i}{NC}$, which represents well the tightness of the given resource allocation problem for the homogeneous maxload constraints that we consider, and of course it should be less than 100% for any instance to be feasible. To achieve a certain load ρ given an instance with a source load vector S, we calculate the required C.

Figure 2 shows CDFs for the feasibility index and the robustness index of $\tilde{A}$ for each of the previous three load distribution models, and for three load conditions (ρ=0.5, 0.6, and 0.7). Each CDF resulted from 1000 executions of the RPV algorithm with a different original instance I_0. The CDF curves that are not visible in the graphs are equal to 100% for all instances. Note that even with the Uniform distribution for S, which is the most likely of the three to produce feasible instances, the feasibility index drops below 90% when ρ is 0.7 (Fig.2(a)). For the heavy-tailed Pareto sources, the feasibility is often below 90% even when ρ=0.6 (Fig.2(c)). The feasibility could be even lower if we had simulated non-homogeneous maxload constraints across different links.

Figure 2 shows that the robustness index of $\tilde{A}$ is larger than 90% for all three load conditions, with both the Uniform and Exponential distributions. With the Pareto distribution, on the other hand, the robustness can be lower than 90% in 10-20% of the cases, but rarely lower than 80% (Fig.2(f)). Note that a higher load ρ does not necessarily mean a lower robustness index. The reason is that the latter is conditioned on



(a) Feasibility, Uniform (b) Feasibility, Exp. (c) Feasibility, Pareto

(d) Robustness, Uniform (e) Robustness, Exp. (f) Robustness, Pareto

Fig. 2. Feasibility and robustness for three source load distributions and load conditions. Note that the CDF curves that are not visible are equal to 100% for all instances

feasible instances only. It can happen that even though the feasibility index is low, a large number of padding vectors are acceptable for the few feasible instances.

The overall conclusion from these results is that RPV can produce a robust padding vector $\tilde{A}$, in the sense that that vector will be acceptable for the actual instance I_a with a probability of more than 80-90%, if the latter is feasible. For I_a to be feasible with a high probability however, say more than 90%, the load metric ρ should be below 50-70%, depending on the distribution of S, at least for the homogeneous maxload constraints that we simulated here.

6 Conclusions

INITE is challenging mostly because it requires that an AS is able to affect BGP routing decisions in remote ASes. ISPs have been using AS-Path prepending to control the flow of ingress traffic. Prepending is widely viewed, however, as an ad-hoc technique, and it has not received much attention in the related literature. In this work, we took a first step of exploring the use of prepending in a more algorithmic framework, and its potential and limitations. Our main contribution is a polynomial-time algorithm (OPV) that can determine the optimal padding vector given the maximum load constraints of ingress links. Even though OPV relies on accurate parameters that can be only roughly estimated in practice, it is still important because it provides the best-case scenario for the effectiveness of prepending if all the required information is available. On a more practical level, the contribution of this paper is to describe how to apply prepending in a robust manner, considering that the AS-Path length information may be subject to estimation errors, and the tie-breaking behavior is unknown. Interestingly, our simulations show that it is possible to determine an acceptable padding vector even in that case, as long as the maximum load constraints are not too tight.

References

1. Fortz, B., Rexford, J., Thorup, M.: Traffic engineering with traditional IP routing protocols. IEEE Communications Magazine (2002)
2. Quoitin, B., Uhlig, S., Pelsser, C., Swinnen, L., Bonaventure, O.: Interdomain traffic engineering with BGP. IEEE Communications Magazine (2003)
3. Feamster, N., Borkenhagen, J., Rexford, J.: Guidelines for interdomain traffic engineering. In: ACM SIGCOMM Computer Communications Review. (2003)
4. Stewart, J.W.: BGP4: Interdomain routing in the Internet. Addison Wesley Longman, Inc. (1999)
5. Gao, R., Dovrolis, C., Zegura, E.: Interdomain ingress traffic engineering through optimized AS-path prepending. Technical Report GIT-CC-05-02, College of Computing, Georgia Tech (2005)
6. Uhlig, S., Bonaventure, O.: Implications of interdomain traffic characteristics on traffic engineering. European Transactions on Telecommunications (2002)
7. Uhlig, S., Magnin, V., Bonaventure, O., Rapier, C., Deri, L.: Implications of the topological properties of Internet traffic on traffic engineering. In: 19th ACM Symposium on Applied Computing, Special Track on Computer Networks. (2004)
8. Quoitin, B., Pelsser, C., Bonaventure, O., Uhlig, S.: A performance evaluation of bgp-based traffic engineering. International Journal of Network Management (Wiley) (2004)
9. Quoitin, B., Tandel, S., Uhlig, S., Bonaventure, O.: Using redistribution communities for interdomain traffic engineering. Computer Communications Journal (2004) 355–363
10. Agarwal, S., Griffin, T.G.: BGP proxy community community. http://www.ietf.org/internet-drafts/draft-agarwal-bgp-proxy-community-00.txt (2004)
11. Quoitin, B., Bonaventure, O.: A cooperative approach to interdomain traffic engineering. In: 1st Conference on Next Generation Internet Networks Traffic Engineering. (2005)
12. Agarwal, S., Chuah, C.N., Katz, R.H.: OPCA: Robust interdomain policy routing and traffic control. In: The 6th IEEE Conference on Open Architectures and Network Programming. (2003)
13. Uhlig, S., Bonaventure, O., Quoitin, B.: Interdomain traffic engineering with minimal BGP configurations. In: 18th International Teletraffic Congress. (2003)
14. Bressoud, T., Rastogi, R., Smith, M.: Optimal configuration for BGP route selection. In: IEEE INFOCOM. (2003)
15. Subramanian, L., Agarwal, S., Rexford, J., Katz, R.H.: Characterizing the Internet hierarchy from multiple vantage points. In: IEEE INFOCOM. (2002)
16. Mao, Z.M., Rexford, J., Wang, J., Katz, R.H.: Towards an accurate as-level traceroute tool. In: ACM SIGCOMM. (2003)
17. Mao, Z.M., Johnson, D., Rexford, J., Wang, J., Katz, R.H.: Scalable and accurate identification of as-level forwarding paths. In: IEEE INFOCOM. (2004)

Describing and Simulating Internet Routes

Jérémie Leguay[1], Matthieu Latapy[2], Timur Friedman[1], and Kavé Salamatian[1]

[1] LIP6 – CNRS and Université Pierre et Marie Curie,
8, rue du Capitaine Scott, 75015 Paris, France
{jeremie.leguay, timur.friedman, kave.salamatian}@lip6.fr
[2] LIAFA – CNRS and Université Denis Diderot,
2, place Jussieu, 75005 Paris, France
latapy@liafa.jussieu.fr

Abstract. This paper introduces relevant statistics for the description of routes in the internet, seen as a graph at the interface level. Based on the observed properties, we propose and evaluate methods for generating artificial routes suitable for simulation purposes. The work in this paper is based upon a study of over seven million route traces produced by CAIDA's *skitter* infrastructure.

Keywords: Network measurements, graphs, statistical analysis, modeling, simulation.

1 Introduction

Realistic modeling of routes in the internet is a challenge for network simulation. Until now, one has had to choose one of the three following approaches to simulate routes: (1) use the shortest path model, (2) explicitly model the internet hierarchy, and separately simulate inter- and intra-domain routing, or (3) replay routes that have been recorded with a tool like `traceroute` [1]. All of these methods have serious drawbacks.

The first method does not reflect reality: routes do not in general have the same properties as shortest paths, as already pointed out by Paxson [2], because of routing policies [3, 4] mainly at the autonomous system (AS) level. As described in detail recently by Spring et al. [3], and earlier by Tangmunarunkit et al. [5, 4], this often induces a lengthening of paths, or *path inflation*, compared to shortest paths. The second method is limited by our ability to explicitly simulate the internet hierarchy. Much work [6, 7] has been done in order to model the internet graph, and much progress has been made, but today's topology generators are still capable of being highly inaccurate in capturing some parameters while they strive to adhere to others. (See, for instance, the findings in Li et al.'s SIGCOMM 2004 paper [8].) Then, even if one is satisfied with the quality of the topology simulation, there is the question of simulating dynamic inter- and intra-domain routing. A non-negligible programming effort is required if the choice is made not to use a simulator, such as *ns* [9], that has these algorithms built in. Finally, the third method is not suitable if routes from a large number of sources are to be simulated. Today's route tracing systems employ at most a few hundred sources. CAIDA's *skitter* [10, 11] infrastructure, for instance, produces an extensive graph suitable for simulations, but it is based on routes from just 30 sources.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 659–670, 2005.

Note that despite its well known drawbacks, and because of the lack of more accurate models, the shortest path model is generally used. Examples from recent years include Lakhina et al.'s Infocom 2003 paper [12], Barford et al.'s SIGCOMM 2002 paper [7], Riley et al.'s MASCOTS 2000 paper [13], and Guillaume et al.'s Infocom 2005 paper [14]. The ns network simulator documentation proposes simulating routes by shortest paths as an alternative to simulating routing algorithms [9–Chs. 26, 29].

This paper's principal contribution is a new approach to modeling routes in the internet, one that does not share the drawbacks just described. We suggest using an actual measured graph of the internet topology, such as the graph generated by skitter. From that topology, we suggest choosing sources and destinations as one wishes from the nodes of the graph. Between these sources and destinations, we suggest generating artificial routes with a model chosen to reflect statistical properties of actual routes.

Central to this contribution are two specific models that we propose for artificial route generation: the random deviation model and the node degree model. These models generate routes with relatively inexpensive calculations, and the routes that they generate better reflect the statistical properties of actual routes than does the shortest path model.

The remainder of this paper is organized as follows. Sec. 2 describes the data set that we have used and the context in which our work lies. Sec. 3 proposes a set of statistical properties to describe routes in the internet. Sec. 4 proposes the models we use to simulate routes based on these properties. Sec. 5 evaluates those models, and Sec. 6 concludes the paper.

2 The Framework

The ideal perspective from which to characterize routes in the internet would be from a snapshot of the routing tables of routers throughout the network. Unfortunately, such a snapshot is impossible to obtain on the scale of the entire network. In this section, we describe the alternative that we opted for, and the hypotheses we made.

2.1 The Internet as a Graph

Efforts to map the internet graph take place at two levels. One is the autonomous system (AS) connectivity graph, which can be constructed from BGP announcements (captured for instance by The Oregon Route Views Project [15]). The other is the router and IP graph, which can be obtained using traceroute and similar tools from a number of different points in the network. To our knowledge, skitter, which conducts traceroutes from on the order of 30 servers to on the order of a million destinations, is the most extensive ongoing effort at the IP level.

Neither level is ideally suited to the task of modeling the behavior of routes at the router level. While the AS graph is directly based upon routing information, it is too coarse-grained to capture the details of path inflation. For this study, we therefore focussed on the IP and router level.

The main problem with this level is that what one actually sees is the graph of IP interfaces, while the graph of routers is more relevant. One single node in the router graph appears as several separate nodes, one or more for each of its interfaces, in the IP graph. Ideally, then, one would construct the router graph using methods to "disambiguate" IP addresses, such as the alias resolution techniques described by Pansiot et al. [16], and by Govindan et al. [17] for *Mercator*. There are also techniques, such as those used by Spring et al. [18, 19], in *Rocketfuel*, and by Teixeira et al. [20], that take advantage of router and interface naming conventions to infer router-level topology.

We do not use the router graph, however. The disambiguation techniques, as applied for example in the *iffinder* tool from CAIDA [21], do not work by simple inspection of the IP graph; they require active probing, preferably simultaneously with graph discovery. This constraint makes extensive disambiguated router-level graphs much harder to obtain than IP interface graphs. At best, some core network topologies are available in this form thanks to Rocketfuel. But Rocketfuel is untested in stub networks. Finally, it is very difficult to judge the extent to which disambiguation is successful, and incomplete or incorrect disambiguation could introduce unknown biases.

To avoid these difficulties, we have restricted ourselves to the IP graph as obtained from skitter. The resulting caveat is that the graph may not be properly representative of the router graph. This caveat is however mitigated by the fact that the IP graph is a legitimate graph in its own right. As Broido et al. note [22], "interfaces are individual devices, with their own individual processors, memory, buses, and failure modes. It is reasonable to view them as nodes with their own connections." If a simulation does not require the explicit modeling of routers then a graph of interfaces can be perfectly adequate. Even more so since certain parameters, such as route lengths, are preserved. That is to say that a route that has a given length in the router graph has the same length in the corresponding IP graph. (Other parameters, such as node degree, will differ, and it is essential to take this into account when interpreting results.)

2.2 The Data Set

This study uses skitter data from July 2nd 2003. During that day, 23 servers targeting 594,262 destinations with 7,075,189 traceroutes. We merge these traceroutes to produce our IP graph. This graph captures the small-world, clusterized, and scale-free nature of the internet already pointed out in numerous publications, see for instance [23, 24]. In particular, the average distance is approximately 12.54 hops, and the degree distribution is well fitted by a power law of exponent 1.97.

Notice that this graph is necessarily incomplete and biased due in particular to probing from a limited number of sources, to route dynamics, to tunneling and to erroneous or absent responses to traceroute probes. Biases of graphs induced by acquisition through a small number of traceroute monitors have been studied for instance in by Lakhina et al. [12]. However, recent studies by Dall'Asta et al. [25] and Guillaume et al. [14] show that one may be quite confident of the accuracy, using this kind of exploration, of distances and degrees, which are the main properties that we study here. We therefore consider the IP interface graph in this study, and in particular we use the skitter data as it represents the current state of the art in its extent and accuracy.

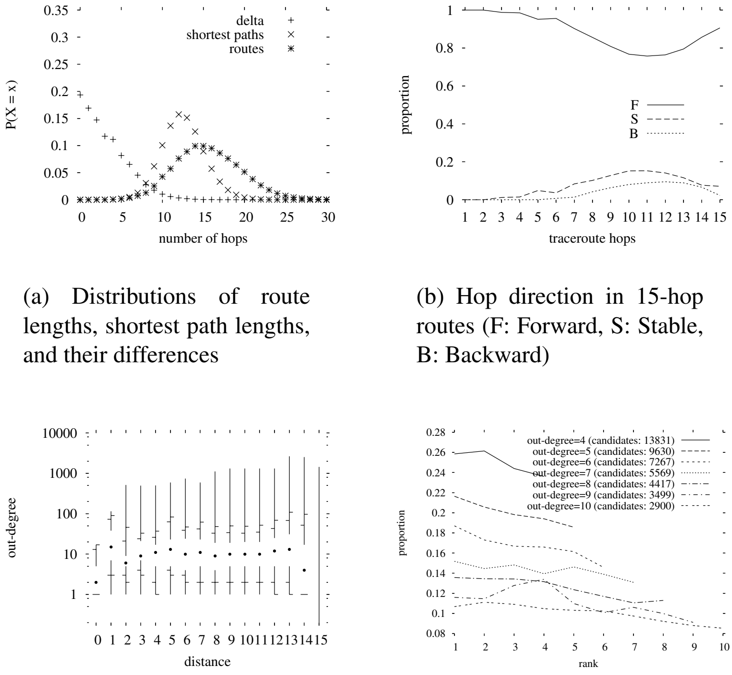
3 Statistical Properties of Routes

This section presents a set of properties for statistical description of internet routes. These properties motivate the models of Sec. 4. Several properties have already been studied in previous work, and the work here serves to evaluate and update them.

3.1 Route Lengths

It is well known that routes are not shortest paths: they are not optimal in general. Fig. 1(a) shows the length distributions of the routes in our data set, and of the corresponding shortest paths. It also shows the distribution of the difference (*delta*) between the length of a route and the corresponding shortest path. The mean length of 15.57 hops for routes in this data set fits closely Paxson's observations [2] on a data set from nine years prior. The shortest paths have a mean length of 12.55 hops (11.4 hops if the graph is considered to be undirected).

The delta distribution confirms Tangmunarunkit et al.'s observation [5, 4], mentioned at the beginning of this paper, that roughly 80% of routes are not shortest paths.



(a) Distributions of route lengths, shortest path lengths, and their differences

(b) Hop direction in 15-hop routes (F: Forward, S: Stable, B: Backward)

(c) Quantiles for the out-degree of nodes along routes of length 15. (See Footnote 1 for details.)

(d) Choice of next hop as a function of its degree ranking. Starting from nodes of degree 4 through 10

Fig. 1. Statistical properties of internet routes

In this data set, 19.34% of routes are shortest paths. Moreover, since the data is incomplete, there are undiscovered links, which implies that 19.34% is an overestimate.

3.2 Hop Direction

When a packet travels from one router to another, it may move closer to its destination, but also it may move farther, or it may move to an interface that is at the same distance from the destination as one it just left. Likewise, the distance from the source may increase, decrease, or stay constant. We will call these behaviors the *hop direction*, considered with respect to either the destination or the source. In principle, a hop should always increase the distance from the source and decrease the distance to the destination; in such cases, the route is a shortest path. Note that hop directions in the router graph can be observed directly in the interface graph, since distances are preserved between the two graphs.

We determine hop direction by computing the shortest path from each traceroute source to all other nodes, using breadth-first search. This is feasible due to the small number of sources. It would also be natural to look at hop direction with respect to the destinations but, since they are much more numerous, it is computationally expensive.

We found that 87.3% of hops go forwards, 4.6% go backwards, and 8.1% remain at the same distance from the source (we call these *stable* hops). More precisely, Fig. 1(b) shows the portion of forward, backward, and stable hops at each hop distance for routes of 15 hops (the most numerous ones). Note that, as one would expect, the first and last few hops are generally forward because there are few alternatives. On the contrary, in the core of the network a significant proportion of the hops (more than one third) do not go closer to the destination. This type of behavior has already been described in the literature as the product of policy-based routing in the core of the internet. As Tangmunarunkit et al. [5, 4] note, such behavior may be induced by load balancing, commercial considerations, etc.

3.3 Degree Evolution Along a Route

Recent work has shown that many real-world complex networks tend to have very heterogeneous degrees, well fitted by power laws. This is in particular true for the internet, as observed by Faloutsos et al. [23] and others. Moreover, most of the short paths between pairs of nodes in these networks tend to pass through the highest degree nodes. Actually, almost all paths (not only short ones) tend to pass through these nodes, which make them essential for network connectivity, see for instance [26, 27, 28, 29, 30].

These observations lead us to ask how the node degree evolves along a route. If routes tend to pass through high degree nodes, where do they do so, and what degree nodes do they encounter? Furthermore, does this tendency to pass through high degree nodes imply that, when a choice exists between next hops, the next hop that leads to the highest degree node is generally chosen?

Fig. 1(c) shows[1] how node degree evolves for routes of length 15. It reveals that a typical route does not pass through the highest degree nodes, though a certain number

[1] In Fig. 1(c), dots indicate the median. Vertical lines run from the min to Q1 and from Q3 to the max. Tick marks indicate the 5th, 10th, 90th and 95th percentiles.

of routes do pass through some very high degree nodes. There is a peak in median out-degree observable at distance 1. The median falls at distance 2, rises again, and then stays fairly flat out to distance 13, with a median degree of about 10. This leads us to the following interpretation: the hosts have low degree, they are connected at their first hop router to relatively high degree nodes which play the role of access points, and then packets are routed in a core network where the degree (typically 10) does not depend much on the distance from the source or from the destination.

Can one observe a simple local rule governing degree evolution? In particular, if there is a choice of next hop interface along a route, is there a correlation between the degree rank of an interface and its probability of being chosen? For instance, are higher degree interfaces chosen preferentially over lower degree ones? Note that such a rule could be perfectly compatible with the observed flat degree evolution.

Fig. 1(d) plots the probability that a packet travels to an interface's i-th ranked neighbor, where the neighbors are ranked from highest out-degree to lowest. An interface's neighbors are its possible next hops in the directed graph. In order to preserve the greatest detail in this middle range, the figure does not show curves for degrees 2 or 3, or above 10, but the curves shown are typical. One can see a general bias towards higher degree nodes, though this bias is rather small, and sometimes is reversed.

4 Route Models

The previous section provides a set of simple statistical tools to capture some properties of routes in the internet. We now propose three simple models (only two of which we eventually retain) designed to capture these features. Each model is based upon one statistical property studied in the previous section. Our approach is to model a property in a very simple way and then use other statistics to validate or invalidate the model.

Whereas our study of route properties was in the context of the directed graphs produced by traceroute, the models in this section are proposed for undirected graphs. The graphs available for simulation purposes, notably those produced by topology generators representing the router-level topology, are typically undirected graphs. Therefore, our models must be suitable for use in this context.

4.1 Path Length Model

The path length model is the simplest and the most obvious one conceptually, but it proves to be unusable in practice. The model aims at producing routes of the same lengths as real ones. As discussed in Sec. 3, a real route length typically exceeds that of the shortest known path by some small integer value $\delta \geqslant 0$.

In order to construct a route from a source s to a destination d, the path length model first computes the length ℓ of a shortest path from s to d. Then it samples a deviation δ from a distribution such as the one shown in Fig. 1(a), and a route is generated by choosing a path at random from s to d among the ones which are loop-free and have length $\ell + \delta$. This ensures that the difference between shortest path lengths and actual route lengths will be captured by the model.

To choose such a path at random implies however that one must construct all of the loop-free paths of length $\ell + \delta$ from s to d. In practice, the computation required to generate this number of paths may be prohibitive, since even in simple cases it is exponential in $\ell + \delta$. For example, in trying to generate all paths of length 21 between a pair of nodes in the skitter graph, we enumerated 1,206,525 possible paths. Therefore, despite its simplicity, we will not consider this model further.

4.2 Random Deviation Model

The random deviation model is based upon the idea that a route usually follows a shortest path, but might occasionally deviate from it. Our model uses a single parameter, p, the probability at any point of deviating from the current shortest path to the destination, if such a deviation is possible. We tuned the value of p to generate routes of realistic length. For the undirected version of the skitter graph, we found $p = 0.2$ to work well.

A random deviation route from source s to destination d is therefore based upon a shortest path u from s to d. At each hop, with probability $1 - p$, the route continues along u. But with probability p it will, if possible, deviate off u to another path. A deviation from current node x to a neighboring node y is deemed possible only if there is a shortest path w from y to d that does not pass through x. Should there be a deviation, the route continues along w to d (unless another deviation should occur). The model is precisely described by Algorithm 1.

Note that large numbers of routes to a destination d can be efficiently generated with the random deviation model once a shortest path tree rooted at d has been computed.

Algorithm 1: rand_dev_route (G, s, d, p)

```
Input    : A network $G$, a source $s$, a destination $d$, a deviation probability $p$.
Output   : An artificial route $v$ from $s$ to $d$ in $G$, following the random deviation model.
Function : sp$(x, y)$ returns the set of all the shortest paths from $x$ to $y$ in $G$.
begin
    $u \leftarrow$ random element of sp$(s, d)$;
    $v \leftarrow$ empty list;
    copy the first element of $u$ to the end of $v$;
    remove it from $u$;
    while the last element of $v$ is not $d$ do
        if $\mathrm{rand}[0, 1] \leqslant p$ then
            $C \leftarrow$ set of all the shortest paths from any neighbor of $v$ to $d$;
            Remove from $C$ the paths containing the last element of $v$;
            if $C \neq \emptyset$ then
                $u \leftarrow$ random element of $C$;
        copy the first element of $u$ to the end of $v$;
        remove it from $u$;
    return $v$;
end
```

4.3 Node Degree Model

Several previous authors [31, 27, 32] have tried to use the heterogeneity of node degrees to compute short paths in complex networks. The basic idea is that a path which goes preferentially towards high degree nodes tends to see most nodes very rapidly (a node is considered to be seen when the path passes through one of its neighbors).

The node degree model is based upon a similar approach, as follows. Two paths are computed, one starting from the source and the other from the destination. The

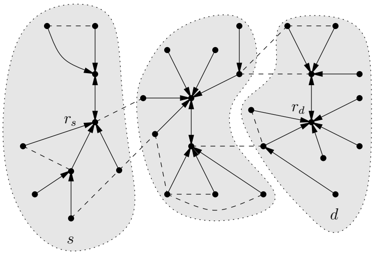


Fig. 2. The node degree model: example

next node on the path is always the highest degree neighbor of the current node. The computation terminates when we reach a situation where a node is the highest degree neighbor of its own highest degree neighbor. One can show that this is the only kind of loop can occur. Then, one of two cases applies: either the two paths have met at a node, or they have not. In the first case, the route produced by the model is the discovered path (both paths are truncated at the meet up node, and are merged). In the second case, we compute a shortest path between the two loops, and then obtain the route by merging the two paths and this shortest path, removing any loops.

In work proceeding in parallel with this paper [32], the node degree model is shown to be an efficient way to compute short paths in complex networks in practice: the obtained paths are very close to shortest ones. Moreover, the computation of the tree-like structure where each node points to its highest degree neighbor is very simple and only has to be processed once. Likewise, the shortest paths between a small number of loops are computed only once. The overall model is described in Algorithm 2.

```
Algorithm 2: node_deg_route ($G$, $s$, $d$)
Input    : A network $G$, a source $s$, a destination $d$.
Output   : An artificial route $v$ from $s$ to $d$ in $G$, following the node degree route model.
Function: reverse($p$): returns the path obtained by reading $p$ from the end to the beginning.
          climb_degrees($G$, $v$): returns the path in $G$ obtained from $v$ by going to the highest degree neighbor at
                                   each hop, until it loops.
begin
    $p_s \leftarrow$ climb_degrees ($G$, $s$);
    $p_d \leftarrow$ climb_degrees ($G$, $d$);
    if $p_s$ and $p_d$ meet up then
        let $u$ be the first node they have in common;
        remove from $p_s$ all the nodes after $u$;
        remove from $p_d$ all the nodes after $u$;
        $p \leftarrow (p_s, \mathrm{reverse}(p_d))$;
        return $p$;
    $q \leftarrow$ random shortest path from the last node of $p_s$ to the one of $p_d$;
    $p \leftarrow (p_s, q, \mathrm{reverse}(p_d))$;
    remove loops from $p$;
    return $p$;
end
```

Fig. 2 is an example. There are three tree-like structures (the shaded areas). The source s belongs to the leftmost one, which is rooted at r_s, and the destination d to the rightmost one, with root at r_d. Each directed link goes from one node to its highest degree neighbor (the dotted lines are links which do not satisfy this). When one wants to build a route from s to d according to the node degree model, one first finds the path from s to r_s, and the one from d to r_d. One then has to compute a shortest path from r_s to r_d, which has length 5 in this example. The final route is obtained by merging these paths, and then removing the loops (which leads to the removal of a link, in our example). It has length 7 (while the shortest path has length 6).

5 Evaluation

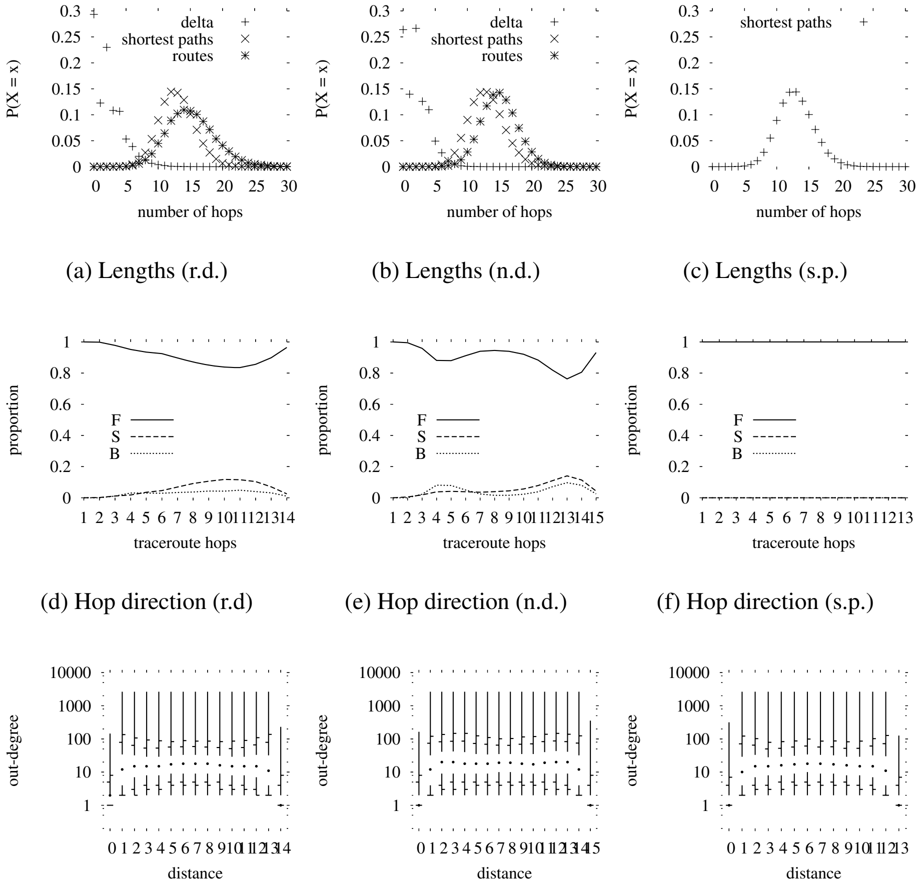
This section compares the performance of the random deviation and node degree models to that of the shortest path model. We use undirected version of the skitter graph described in Sec. 2.2, considered as an undirected graph. For each model, we chose at least 60,000 (source, destination) pairs at random from amongst the nodes of the graph and generated an artificial route from the source to the destination. We compute the same statistics on these routes as we had computed for actual routes in Sec. 3.

Fig. 3 shows the statistics for each model. We judge the quality of a model by how well its statistics mirror those for actual routes, shown in Fig. 1.

Comparing the route length distributions, we find that both models generate distributions that are symmetric, average somewhat higher than the shortest path distribution, and have tails similar to the actual route length distribution shown in Fig. 1(a). Mean route length is 15.15 for the random deviation model and it is 14.96 for the node degree model, whereas the mean shortest path is 12.93. (Note that, on the undirected skitter graph, shortest paths between random sources and destinations are longer on average than those between skitter sources and destinations, for which we had computed an average route length of 11.21.)

Lengths of paths generated with the node degree model tail off somewhat quicker than in reality (approaching zero closer to length 20 than length 25), but the degree of fidelity is nonetheless remarkable given that the length distributions are not explicitly part of the model. The random deviation model generates more routes that are shortest paths than in reality (roughly 30% compared to roughly 20%), whereas the node degree model generates somewhat fewer (roughly 26%). As is already known, the shortest path model does not capture the length properties.

Looking at the hop directions for the most frequent route length, we found that the curves for the random deviation model better match the shapes of the curves for real routes shown in Fig. 1(b). Hops are mostly forward near the source, but dip to around 80% roughly ten hops out (whereas in reality the portion of forward hops dips to around 80% at eleven or twelve hops out). This is in marked contrast to hop directions produced by the node degree model because forward hops dip much sooner and a bit less steadily. But overall portions of forward, stable, and backward hops closely match reality for both models: 89% forward, 7% stable, and 4% backward for the random deviation model, and 90% forward, 6% stable, and 4% backward for the node degree model, compared to 87% forward, 8% stable, and 5% backward for true routes.



(a) Lengths (r.d.) (b) Lengths (n.d.) (c) Lengths (s.p.)

(d) Hop direction (r.d) (e) Hop direction (n.d.) (f) Hop direction (s.p.)

(g) Out-degree (r.d.) (h) Out-degree (n.d.) (i) Out-degree (s.p.)

Fig. 3. Experiments using the *random deviation model* (left), the *node degree model* (center), and the *shortest path model* on the undirected skitter graph using sources and destinations chosen at random from amongst all the nodes in the graph

The shortest path model fails to capture these proportions since all of its links are forward.

The node degree model does a better job than the random deviation model in capturing the evolution of the out-degree close to a route's source. Routes generated with this model show the peak in the out-degree before settling down to a median around 20 that we noticed in Fig. 1(c), though the peak is reached at distance 2 rather than at the first hop router. The random deviation model and the shortest path model also have a median around 20, but they arrive there through a smooth increase, with no clear peak.

Based upon this comparison to real routes, we can state that the random deviation and node degree models do a reasonable job of emulation, though each model captures some aspects better than others, and their strengths are different. Both models clearly out-perform the shortest path model.

6 Conclusion and Future Work

The main contribution of this paper has been to propose a new alternative for the simulation of routes in the internet: the use of simple models that capture non-trivial statistical properties of routes. The models proposed here have been found to reproduce a number of aspects of true internet routes, though neither fully captures all of the characteristics. Our goal was to introduce simple models that could serve as alternatives to the clearly unrealistic shortest path model. No model can be fully faithful to reality, and the key is to understand in what ways it is a true representation, and in what ways it diverges. Future work along these lines might include the development of models that explicitly incorporate some additional characteristics, such as the clustering coefficient. Other work might involve studying whether certain variants on the models, such as a hybrid of the random deviation and node degree approaches, would be more like real routes. Any such work must keep in mind the desirability of keeping the models conceptually simple, easy to implement, and computationally tractable.

Another area into which this work could be extended would be to capture something of the dynamics of internet routes. There are effectively random choices to be made in both the random deviation model (clearly) and the node degree model (when it comes to choosing among two or more neighbors of highest degree, or choosing a shortest path between two trees) but we have not touched on the timing of that variation.

Acknowledgments. We thank kc claffy and the staff at CAIDA for making the skitter data available to us. We also thank Mark Crovella and Miriam Sofronia for helpful comments. This work was supported in part by the RNRT through the Metropolis and MetroSec projects.

References

1. Jacobsen, V.: traceroute (1989) `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`. Also see “NANOG traceroute,” `ftp://ftp.login.com/pub/software/traceroute/`.
2. Paxson, V.: End-to-end routing behavior in the Internet. IEEE/ACM Trans. on Networking **5** (1997) 601–615 See also Proc. ACM SIGCOMM 1996.
3. Spring, N., Mahajan, R., Anderson, T.: Quantifying the causes of path inflation. In: Proc. ACM SIGCOMM. (2003)
4. Tangmunarunkit, H., Govindan, R., Shenker, S.: Internet path inflation due to policy routing. In: Proc. SPIE ITCom. (2001)
5. Tangmunarunkit, H., Govindan, R., Shenker, S., Estrin, D.: The impact of routing policy on internet paths. In: Proc. IEEE Infocom. (2001)
6. Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., Willinger, W.: Network topology generators: Degree-based vs. structural. In: Proc. ACM SIGCOMM. (2002)
7. Barford, P., Bestavros, A., Byers, J., Crovella, M.: On the marginal utility of network topology measurements. In: Proc. Internet Measurement Workshop (IMW). (2001)

8. Li, L., Alderson, D., Willinger, W., Doyle, J.: A first-principles approach to understanding the internet's router-level topology. In: Proc. ACM SIGCOMM. (2004)
9. Fall, K., Varadhan, K.: The ns manual (2003)
10. Huffaker, B., Plummer, D., Moore, D., claffy, k.: Topology discovery by active probing. In: Proc. Symposium on Applications and the Internet (SAINT). (2002)
11. CAIDA: (skitter) a tool for actively probing the Internet, `http://www.caida.org/tools/measurement/skitter/`.
12. Lakhina, A., Byers, J., Crovella, M., Xie, P.: Sampling biases in IP topology measurements. In: Proc. IEEE Infocom. (2003)
13. Riley, G.F., Ammar, M.H., Fujimoto, R.: Stateless routing in network simulations. In: MASCOTS. (2000) 524–531
14. Guillaume, J.L., Latapy, M.: Relevance of massively distributed explorations of the Internet: Simulation results. In: Proc. IEEE Infocom. (2005)
15. Meyer, D.: (University of Oregon Route Views Project) `http://www.antc.uoregon.edu/route-views/`.
16. Pansiot, J.J., Grad, D.: On routes and multicast trees in the Internet. ACM SIGCOMM Computer Communication Review **28** (1998) 41–50
17. Govindan, R., Tangmunarunkit, H.: Heuristics for internet map discovery. In: Proc. IEEE Infocom. (2000)
18. Spring, N., Mahajan, R., Wetherall, D.: Measuring ISP topologies with Rocketfuel. In: Proc. ACM SIGCOMM. (2002)
19. Spring, N., Dontcheva, M., Rodrig, M., Wetherall., D.: How to resolve IP aliases. Tech. Report 04-05-04, Washington Univ. Computer Sci. (2004)
20. Teixeira, R., Marzullo, K., Savage, S., Voelker, G.: In search of path diversity in ISP networks. In: Proc. Internet Measurement Conference (IMC). (2003)
21. Keys, K.: (iffinder) a tool for mapping interfaces to routers, `http://www.caida.org/tools/measurement/iffinder/`(restricted access).
22. Broido, A., claffy, k.: Internet topology: Connectivity of IP graphs. In: Proc. SPIE International Symposium on Convergence of IT and Communication. (2001)
23. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: Proc. ACM SIGCOMM. (1999)
24. Vazquez, A., Pastor-Satorras, R., Vespignani, A.: Large-scale topological and dynamical properties of the Internet. Phys. Rev. E 65, 066130 (2002)
25. Dall'Asta, L., Alvarez-Hamelin, I., Barrat, A., Vasquez, A., Vespignani, A.: A statistical approach to the traceroute-like exploration of networks: theory and simulations. In: Proc. 1st Int'l. Conf. on Combinatorial and Algorithmic Aspects of Networks (CAAN). (2004)
26. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance in complex networks. Nature **406** (2000) 378–382
27. Kim, B.J., Yoon, C.N., Han, S.K., Jeong, H.: Path finding strategies in scale-free networks. Phys. Rev. E 65, 027103 (2002)
28. Cohen, R., Erez, K., ben Avraham, D., Havlin, S.: Breakdown of the internet under intentional attack. Phys. Rev. Lett. **86** (2001) 3682–3685
29. Cohen, R., Erez, K., ben Avraham, D., Havlin, S.: Resilience of the internet to random breakdown. Phys. Rev. Lett. **85** (2000) 4626
30. Callaway, D., Newman, M., Strogatz, S., Watts, D.: Network robustness and fragility: Percolation on random graphs. Phys. Rev. Lett. **85** (2000) 5468–5471
31. Walsh, T.: Search in a small world. In: Proc. IJCAI. (1999)
32. Bampis, E., Latapy, M., Pascual, F.: Computing short paths in scale-free networks. (2004) preprint.

A Growth-Based Address Allocation Scheme for IPv6

Mei Wang

Stanford University, Stanford, CA 94305, USA
wmei@stanford.edu

Abstract. IP address allocation policies significantly impact the Internet infrastructure, affecting many parties such as router manufacturers, ISPs, and end users. An address allocation policy can also directly affect the performance of the Internet. For example, address fragmentation, a key problem in IPv4, degrades address lookup performance in routers. Thus, a well-designed address allocation policy needs to minimize fragmentation while using the address space efficiently.

This paper attempts to quantify the performance of address allocation policies by modeling key features that lead to fragmentation and inefficient address space usage. Our main contributions are: (i) we identify a drawback of the current IPv6 address allocation policy, which treats all entities uniformly, (ii) we propose a scheme that takes future growth rate into account for allocations, and (iii) an analytical model for measuring the efficiency of allocation schemes, allowing us to quantify the improvement our proposal offers over the current scheme. We believe that a quantitative study of allocation policies is timely since IPv6 address allocation is just beginning in earnest.

1 Introduction

IP address lookup is a key element of packet processing in Internet routers. The performance of lookup algorithms is largely impacted by the size and structure of routing tables. Recent research has shown that address allocation policies significantly affect the structure and growth of routing tables and, hence, the performance of lookup algorithms [4, 5, 7]. Therefore, it is natural to ask: How should one design an address allocation scheme that will lead to well-structured routing tables? A timely answer to this question will help the address allocation practice for IPv6, because with its larger address space (due to 128-bit addresses) IPv6 poses challenging problems to the performance of lookup algorithms [1, 2].

Address space fragmentation, a phenomenon that causes one entity (say an ISP) to have multiple non-contiguous address blocks or prefixes instead of a single prefix, is one cause of bad routing table structures. This makes address lookup algorithms, which are longest-matching-prefix based, to perform poorly. While there are several reasons why address fragmentation occurs, one key factor is a poor address allocation policy. This is because addresses are allocated to an

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 671–683, 2005.

entity on an “as-needed” basis. Thus, an entity whose size experiences a large growth might end up with non-contiguous address blocks.

Avoiding, or minimizing, address space fragmentation is a major goal of an allocation policy. Another major goal is address space conservation [8]; that is not overallocating addresses to an entity which may not use the entire allocation. Given IPv6’s large address space, it is tempting to overallocate so as to minimize the chance of address fragmentation. However, as demonstrated by many examples in the history of computing and networking, it is important to guard against this temptation since a large supply of a resource invariably invites its increased usage (some of which is even hard to foresee). Therefore, we may summarize the goals of a good address allocation policy as the minimization of address fragmentation while ensuring an efficient use of the address space.

This paper provides a quantitative model and an accompanying analysis of allocation schemes, rating them on the degree of address aggregation (as opposed to fragmentation) and conservation they achieve. The current allocation policy suggests a bisection algorithm (described in more detail later) for maximizing address aggregation [15]. However, it does not take into account the potential future growth of an entity while making the initial allocation. Here we propose a scheme that dynamically partitions the address space according to the growth rate of each entity. Using a theoretical model and via simulations we find that, compared to the current bisection method, our growth-based scheme significantly reduces address fragmentation and improves the efficiency of address usage.

The rest of this paper is organized as follows: Section 2 presents the background information on the current IPv4 and IPv6 allocation policies. Two allocation algorithms, bisection and growth-based schemes, are described in Section 3. Theoretical models and analysis are presented in Section 4 followed by simulation results presented in Section 5. Section 6 presents the conclusion.

2 Background

IP addresses are allocated hierarchically. The size of the address blocks decreases as the level of the hierarchy increases. With reference to Figure 1, the addresses at the top of the hierarchy are controlled by the Internet Assigned Numbers Authority (IANA). IANA allocates large address blocks to each of the five Regional Internet Registries (RIR) serving the North American, European, Asian, African, and the Latin American and the Caribbean regions. The regional registries divide up these large address blocks into medium blocks to allocate to Local Internet Registries (LIRs), consisting mainly of ISPs. The ISPs further assign small address blocks to their customers, including companies, universities, smaller ISPs, etc.

2.1 Allocation Policies

In this paper, we focus on global unicast address allocation which consumes most of today’s address space. Each block of addresses is represented by a prefix which is denoted by prefix-value/prefix-length, e.g., 5.0.0.0/8 in IPv4 and 2000::/3

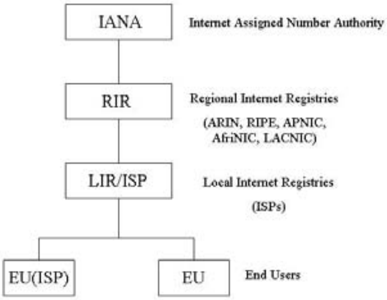

IANA
Internet Assigned Number Authority
RIR
Regional Internet Registries
(ARIN, RIPE, APNIC, AfriNIC, LACNIC)
LIR/ISP
Local Internet Registries
(ISPs)
EU(ISP)
EU
End Users

Fig. 1. IP address allocation hierarchy

in IPv6 [3]. The prefix length specifies the number of significant bits in the prefix value. For IPv4 with 32-bit total address length, an allocation of 5.0.0.0/8 represents an address block size of 2^{24} containing addresses all begin with 5 as the first byte in the address values. The smaller the prefix length, the larger the block size.

The current IPv4 policies are [8-11]: IANA allocates to RIRs in the unit of /8. Different RIRs adopt their own policies for allocations to LIR/ISPs with unit sizes varying from /10 to /20. The sizes assigned to end users by each ISP also vary greatly. Due to historical allocation schemes, fragmentation is a common problem in IPv4 [7]: one ISP is often left with multiple prefixes.

For IPv6, although there are 128 bits in the IP address, the last 64 bits are assigned to interface ID [2], e.g., an ethernet's MAC address. Thus, address allocation only considers the top 64 bits. The allocation unit from IANA to RIRs is /23 [13]. RIRs are coordinating to have a common policy for allocation to LIR/ISPs [14]. Currently, the minimum initial allocation unit is /32. There is a proposal [15] from major RIRs recommending a Common Address Pool (CAP) in IANA for all RIRs, instead of each RIR keeping a separate pool of addresses for allocation. To end users, unlike IPv4, IPv6 is generally assigned in fixed amounts (/48).

Allocation policies vary for different registries and different layers of the hierarchy. At present, documents on allocation policies mainly specify the size of prefixes and the criteria for allocation. Few specific procedures exist on allocation algorithms and how the address space should be partitioned. A good allocation algorithm can be applied to any layer in the allocation hierarchy and any registry. It can be combined with the allocation policy to provide efficient usage of address space and preserve address aggregation.

3 Allocation Algorithms

For a given address pool with the address length of N bits, the total address space of this pool is 2^N. This space can be represented by an address line, ranging from 0 to 2^N-1. Each allocation is a block of addresses and can be shown as a section on the address line. The starting location of the address block is labeled by the prefix value. The size of the address block, also the length of this section on the address line, is 2^{N-l}, where l is the prefix length.

For an address provider with a prefix length $/P$ and its customer with prefix length $/l$, their address spaces are 2^{N-P} and 2^{N-l}, respectively. For IPv6, N=64 if we use the first 64 bits as the total address space for allocation. An ISP with /32 can have a total of $2^{(64-32)}/2^{(64-48)}=2^{16}$ customers with /48 if none of them out-grow their address space. If the smallest address block size allocated by a provider with $/P$ has a prefix length $/l_{min}$, we can treat this block size as a base unit. The problem is equivalent to allocating an address space of $2^{l_{min}-P}$ with each customer occupying a minimum space of 1.

When a customer grows out of its initially allocated address space, the provider can double this customer's space by combining the customer's current address block with the one immediately after it on the address line in order to keep the same prefix value. The prefix length of the customer is reduced by 1. This process can be repeated every time it out-grows its current space. HD ratio [18] can be used for expansion criteria.

A collision occurs when a customer needs to expand its space and a neighboring customer on the address line is already occupying that space or a section of that space. When this occurs, the provider can either find a new location on the address line with a different prefix value or still double the customers space by carving out the section already occupied by the neighbor. Both options lead to fragmentation: non-continuous address blocks for one entity, i.e., more than one prefix representing a single entity in the routing table. Fragmentation reduces lookup and routing efficiency and increases routing table size. Such practice should be avoided whenever it is possible. A third option is to move either one of the two customers to a different address location, which will create much hassle and may not always be viable.

We describe two address allocation schemes in this section: the basic bisection scheme suggested by the registries and our proposed scheme based on the growth rate of each allocation.

3.1 Bisection Scheme

The current address allocation practice is based on the bisection scheme, proposed by registries of North America, Asia, and Europe. Address blocks are allocated bisectionally in the following way: each new address block is allocated by evenly splitting a section on the address line between the existing customer and the new customer. This leaves maximum possible space for potential growth of both customers. An example using this scheme is illustrated in Figure 2. The first four customers are allocated by dividing the total address space equally into four parts as shown in Figure 2(a). When the fifth customer applies for an address block, it is placed to evenly split the section labeled by the arrow in Figure 2(b). The 6th, 7th, and 8th customer will be placed sequentially in the spaces after the 2nd, 3rd, and 4th customer, respectively. Only after the largest empty slots have been exhausted, will new allocations be assigned to bisect the smaller slots at the next level. A similar technique can be found for dynamic memory allocation in Operating Systems and is already used in IPv4.

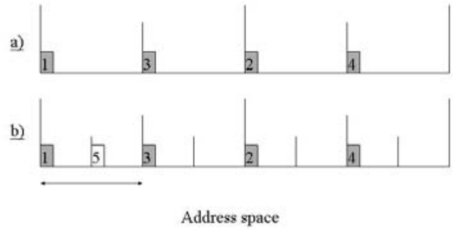


Fig. 2. Bisection scheme

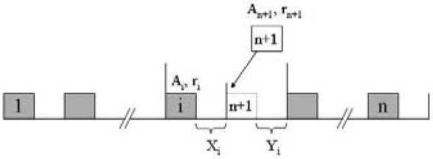


Fig. 3. Procedure to determine address location for the new customer using growth-based scheme

This method uniformly separates the allocated blocks to maximize the spacing between customers. However, different customers are likely to have different growth rates and require different sizes in the address space. Thus this uniform partition may not be the most efficient way to utilize the address space, especially when there are a few very fast-growing customers. These fast-growers can collide with their neighbors and cause address space fragmentation.

3.2 Growth-Based Scheme

To utilize the address space more efficiently and reduce collisions, a modified scheme is proposed to make allocations based on the growth of each customer. When there are n exiting customers, there are at most n possible address location candidates for the $(n+1)^{th}$ customer. Instead of treating each customer equally, as in the bisection scheme, the growth-based scheme evaluates all the options and chooses the location based on the available space sizes as well as the growth of the existing customers and the new customer. For example, one can choose the location for the new customer that maximizes the time before the first collision is projected to take place. To maintain the same prefix value for each address block, the newly allocated block has to start at the middle point of the available address space.

The ability to obtain reasonably accurate growth estimation directly impacts the effectiveness of growth-based algorithms. Since each address provider (a registry, an ISP, or a company) can access the information of its customers' utilization, the estimated growth of each customer can be derived from its growth history and current utilization. Price related incentives can be applied to help collect more accurate estimations from customers. This scheme still works even when the estimation of the growth rate is not accurate or even wrong. Because for each address allocation there is still space left for potential growth. Since we check each available block's growth every time inserting a new comer, the growth rate can be adjusted frequently and adaptively according to any changes. The worst case would be a fast growth entry coming in when the whole address space is about full, then there will be less gain or no gain using this scheme, but this case is rare.

For now, we assume that the growth information is available, known as growth rate. Let

n : the number of existing customers; $n+1$: the new customer;
A_i : i=1,...,n,n+1, allocation for each customer;
r_i : i=1,...,n,n+1, estimated growth rate of each allocation;
X_i : i=1,...,n, empty space behind each allocated block to bisection point;
Y_i : i=1,...,n, the empty space available for newly allocated customer to grow;

as shown in Figure 3. The new location is chosen to maximize the time before the first collision occurs, using:

$$max\{min[t(X_i, r_i), t(Y_i, r_{n+1})], i = 1, ..., n\}. \tag{1}$$

$t(X, r)$ is the time it takes for allocation A with growth rate r to fill up the space X. In other words, the new location is chosen among all available spaces that maximizes the time it takes for either the existing or the new customer to run into the boundary. Function $t(X, r)$ can be of different forms depending on the definition of growth rate r.

4 Theory

To quantitatively study the effects of the allocation schemes, we model the system where new customers keep being allocated in the address space and continue growing in size with various rates. Analytical expressions are derived in this section and are compared to simulations in the next section. In order to obtain closed-form expressions, we treat the address space and the space needed by individual customers as continuous variables in this section. The total allocated address space for IPv4 has been growing exponentially during the last decade, i.e., proportional to the total size of the network [17]. For the rest of this paper, we use the same functional form to represent growth, i.e., exponential increase for both the number of customers and the space needed by individual customers. The methodology developed here can be easily adapted for other functional forms of growth. To compare the performance of different allocation schemes, we compare the total number of customers served and the total address space utilized before the first segmentation takes place.

4.1 Bisection Scheme

We assume that the number of customers grows proportional to the total number of existing customers, i.e., exponential increase with time

$$n(t) = n_0 f(t), \quad f(t) = 2^{v \times t}, \tag{2}$$

where n_0 is the initial number of customers at t=0 and $f(t)$ is the growth function of the number of customers. v is the growth rate. Different forms of growth functions can be used in place of expression $2^{v \times t}$ for other growth patterns.

The bisection scheme attempts to partition the total address space S_{tot} equally among all customers. The amount of space to grow for each customer, $s(t)$, decreases with time as more customers are added to the address space:

$$s(t) = S_{tot}/n(t) = S_{tot} \times 2^{-v \times t}/n_0. \qquad (3)$$

As time progresses, customers can request additional space as they themselves grow in size. Customers can be grouped into different classes according to their growth rates r_i and initial address space l_i^0, where i is the class index (1≤i≤ m, m is the total number of classes). Within each class, the customers are labeled by index j which corresponds to the order of which they are added into the address space. Depending on the time of its entry, $t_{i,j}^0$, the size of the address space a customer requires at a later time t is

$$l_{i,j}(t) = l_i^0 \times 2^{r_i \times (t - t_{i,j}^0)}. \qquad (4)$$

The sequence of incoming customers are considered to be random. The entrance time $t_{i,j}^0$ of the j^{th} customer of class i can be estimated from the following: when the total number of customers at this time multiplied by the probability p_i (the probability that a customer picked at random belongs to class i) equals j, i.e., $n(t_{i,j}^0) \times p_i = j$,

$$t_{i,j}^0 = \frac{1}{v} \times \log_2(\frac{j}{p_i \times n_0}). \qquad (5)$$

A collision takes place when the space needed by a customer becomes greater than the space available for its growth. Since it is the first customer of each class that has the largest size at a given time, the problem of finding the time at which first collision occurs reduces to $t_{min}^c = min \left\{ t_{i,1}^c, \;\; i = 1, ..., m \right\}$. The time of collision of the first customer of class i, $t_{i,1}^c$, is obtained by equating the average space for growth of each customer [Eq.(3)] and the space needed [Eq.(4)]:

$$t_{i,1}^c = \frac{1}{v + r_i} \times \left[-\frac{r_i}{v} \log_2(n_0 \times p_i) + \log_2 \left(\frac{S_{tot}}{n_0 l_{i,1}^0} \right) \right]. \qquad (6)$$

The total number of customers served at the time of the first collision follows from Eq.(2), $n(t_{min}^c) = n_0 \times 2^{v \times t_{min}^c}$.

To obtain the total space used at the time of the first collision, one can sum over the space occupied by existing customers of each class. This can be done by finding the number of customers of each class at t_{min}^c, the time of entry of each customer, and the space needed by each customer at t_{min}^c. To obtain an analytic expression, we approximate the summation by an integral over time,

$$S(t_{min}^c) = \sum_{i=1}^{m} \int_0^{t_{min}^c} l_i^0 2^{r_i (t_{min}^c - t)} p_i \frac{dn(t)}{dt} dt = \sum_{i=1}^{m} n_0 l_i^0 p_i v \frac{2^{v t_{min}^c} - 2^{r t_{min}^c}}{v - r_i}. \qquad (7)$$

Eq.(7) can be understood as follows: $[dn(t)/dt]\,dt$ is the number of new customers added between time t and $t+dt$. Among these, $p_i \times [dn(t)/dt]\,dt$ of them belong to class i that grow to size $l^0_{i,1} \times 2^{r_i \times (t^c_{min}-t)}$ at t^c_{min}.

The results from the analytic method is shown in Figure 4 as a function of the total address space. These results compare well with a simulation to be described in Section 5. The small deviation of the analytical method from the simulation results comes from the fact that the address space is treated as continuous in the analytic method as opposed to integers in the simulation. The same can be said about the results given in Figure 5 for the growth-based scheme.

4.2 Growth-Based Scheme

The growth-based scheme dynamically optimizes the partition in the address space for each incoming customer. We find that it is possible to obtain a closed-form expression for a specific case where there are only two classes of customers: (1) ones with a fixed size without growth and (2) ones that grow and grow at the same rate. In reality, we can categorize most customers into these two groups: slow growth and fast growth. The probabilities of finding a customer belonging to classes 1 and 2 are p_1 and p_2, respectively, with p_1+p_2=1. For simplicity, we assume that customers of both classes require an initial size of 1.

When the first customer of class 2 enters into the address space, there are, on average, a total of $1/p_2$ customers in the address space. The available space for this customer to grow would be roughly $s_{2,1} = S_{tot}p_2$. We make an assumption that the optimum arrangement of the address space would be letting each customer of class 2 (i.e., with growth) keep all its potential growth space obtained at its initial allocation. We will show at the end of this section that this assumption is justified under certain conditions. Following this assumption, when the second customer of class 2 enters, there are $2/p_2$-1 customers partitioning the rest of the space,

$$s_{2,2} = (S_{tot} - s_{2,1})/(2/p_2 - 1) = S_{tot}(p_2(1-p_2))/(2(1-p_2/2)). \qquad (8)$$

Analogously, the j^{th} customer of class 2 gets a growth-space of size

$$s_{2,j} = S_{tot}p_2(1-p_2)^{j-1} \Big/ \left[j \prod_{k=1}^{j} \left(1 - \frac{k-1}{k} p_2\right)\right]. \qquad (9)$$

Since customers of class 1 keep a fixed size, we only need to consider the collisions from customers of class 2. The space that the j^{th} customer of class 2 requires at a later time t is given by $l_{2,j}(t) = 2^{r_2 \times (t - t^0_{2,j})}$, where $t^0_{2,j}$ is the time of its entry and can be obtained using Eq.(5). The time that the j^{th} customer of class 2 exhausts its growth space is when $l_{2,j}(t)$=$s_{2,j}$:

$$t^c_{2,j} = \frac{1}{v} \times \log_2 \frac{j}{n_0 p_2} + \frac{1}{r_2} \times \log_2 \left[S_{tot} \frac{p_2(1-p_2)^{j-1}}{j \prod_{k=1}^{j}\left(1 - \frac{k-1}{k} p_2\right)} \right]. \qquad (10)$$

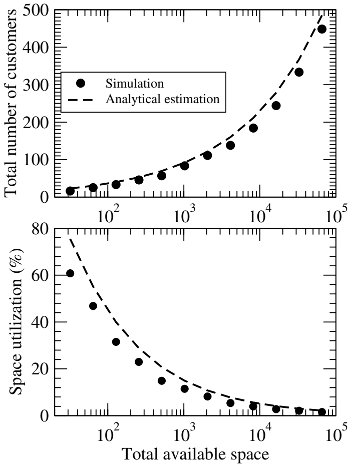


Fig. 4. Bisection scheme: comparison of analytic method with simulation

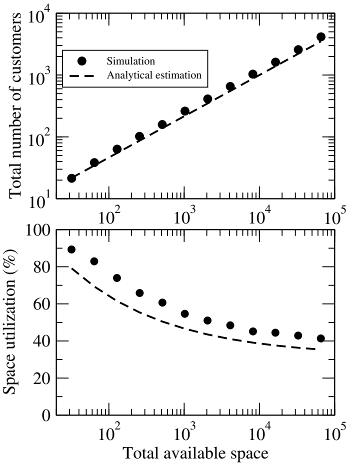


Fig. 5. Growth-based scheme: comparison of analytic method with simulation

To determine which customer of class 2 is the first to exhaust its growth space, we compare the time-to-collision between the $(j+1)^{th}$ and j^{th} customers:

$$\begin{aligned} t^c_{2,j+1} - t^c_{2,j} =& \frac{1}{v} \times \log_2 \frac{j+1}{j} + \frac{1}{r_2} \times \log_2 \left[\frac{j(1-p_2)}{(j+1)\left(1 - \dfrac{j}{j+1} p_2\right)} \right] \\ \approx& (r_2 - v)/(v \times r_2) \times \log_2 (1 + 1/j) \qquad (\text{if } p_2 \ll 1) \\ >& 0 \qquad\qquad\qquad\qquad\qquad (\text{if } r_2 > v), \end{aligned} \tag{11}$$

where the approximate sign holds when the probability of fast-growers (i.e., class 2) is much less than 1. Eq.(11) shows that the time-to-collision increases with the customer number under the condition that the space growth-rate of the growing customers (r_2) is faster than the rate of increase in the total number of customers (v). Our starting assumption, that each customer of class 2 keeps all its potential growth space, is therefore justified as being the optimum choice for the placement of incoming customers under these conditions. Therefore, the first customer of class 2 is the first to hit its boundary of potential growth:

$$t^c_{min} = -1/v \times \log_2 (n_0 p_2) + 1/r_2 \times \log_2 (S_{tot} p_2) . \tag{12}$$

The total number of customers when the first collision takes place can be obtained from Eqs.(12) and (2), $n(t^c_{min}) = p_2^{\frac{v - r_2}{r_2}} \times S_{tot}^{\frac{v}{r_2}}$. The total space utilized follows from Eqs.(12) and (7):

$$S(t^c_{min}) = \frac{v}{r_2 - v} n_0^{1 - \frac{r_2}{v}} \times p_2^{2 - \frac{r_2}{v}} \times S_{tot} + \left(1 - \frac{p_2 r_2}{r_2 - v}\right) \times p_2^{\frac{v - r_2}{r_2}} \times S_{tot}^{\frac{v}{r_2}} . \tag{13}$$

Figure 5 shows that the results using the analytic method agree with those obtained by a simulation technique described in the next section. These results are obtained using the following parameters: p_2=10%, r_2=1/4, v=1/6, n_0=4.

5 Simulation Results

The projected growth rate is essential for the application of the growth-based algorithm. The growth rate can be represented by various functional forms, such as the total projected address space, the probability that extra bits will be needed in the future, the projected size increase for each year, etc. The basis of the theoretical analysis and simulation techniques put forth in this paper can be adapted for the different functional forms or their combinations.

The performance differences on address space conservation and fragmentation between the bisection and growth-based schemes are compared using simulations. Two metrics are used to measure the performance on conservation: the total number of customers served and the percentage of address space allocated before the first collision takes place. This implies that all the customers served have one continuous address block without fragmentation, i.e., every customer has one prefix only. For performance on fragmentation, the percentage of fragmented addresses is measured for different address utilizations.

In the simulation we assume that customers with different growth rates enter the address space at random order. The identical sequence of customers is fed to both schemes for a one-to-one comparison. A variety of different distributions in the customers' growth-rates have been explored with qualitatively similar results. The results for a Gaussian distribution are presented in this paper, which serves as a representative. In these simulations, most of the customers have medium growth rates but there are a few with very fast or very slow growth rates.

Without loss of generality, we denote the minimum block size allocated by a provider as one base unit in the address space. Simulations of 2^N total address space sizes are considered, with N ranging from 5 to 16. The case N=16 corresponds to an ISP with prefix length /32 making allocations with the smallest block sizes with prefix length /48. This is exactly the practice in current IPv6 address allocation.

5.1 Address Space Conservation

Fixed Initial Prefix Length Allocation. The first group of simulations is carried out on fixed initial prefix lengths. According to the current IPv6 allocation policies, the initial assignment is, in almost all cases, of the same initial length for each new customer. It is only when a customer has achieved a certain utilization of the initial address space that it is given a larger block of address space. This policy applies to each layer of the allocation hierarchy with official registries.

Figure 6(a) shows that the growth-based scheme can serve many more customers than the bisection scheme before the first collision. This means more addresses are allocated by the time the first collision happens, thus, the address utilization is higher. The larger the size of the total address space, the more significant the gain is with the growth-based scheme given the identical profile of customers. The same can be said about the percentage of space utilized, as shown in Figure 6(b). For the case with a total address space of 2^{16}, the growth-based scheme gives a 16-fold improvement over the bisection scheme for the number of customers served and a 42-fold increase in terms of space occupancy.

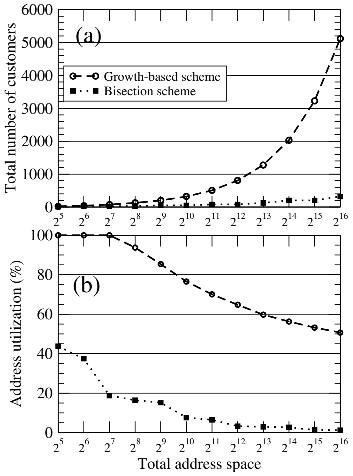


Fig. 6. Simulation results on address space conservation for both bisection and growth-based schemes with fixed initial prefix length allocation

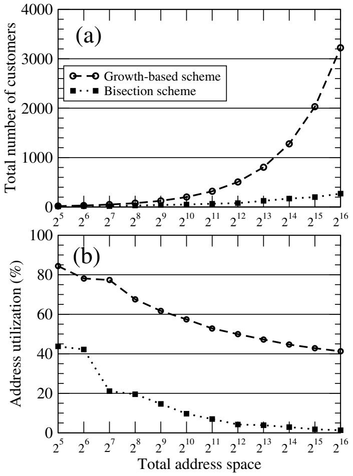


Fig. 7. Simulation results on address space conservation for both bisection and growth-based schemes with variable initial prefix length allocation

The amount of gain of the growth-based scheme over the bisection method depends on the input profile. One factor in the input profile is the order that different customers enter into the address space. Another factor is the growth rate of each customer relative to the rate of increase in the total number of customers. From a large number of experiments we explored over different input configurations, the gains of the growth-based scheme are all very substantial but vary in quantity. In one case the gain is not obvious: if a customer with a very fast growth rate comes in when the space is relatively full and the size of the requested block is of the same order as the size of the empty slot. In this particular case, there is not much time left before this new customer reaches its space limit for growth. This very limited amount of time is sometimes not enough to show a dramatic benefit for the growth-based scheme over the bisection one.

Variable Initial Prefix Length Allocation. Even though the current policies require giving a fixed initial prefix length for each new allocation, it is possible that such policy will be relaxed in the future to allow variable initial allocation lengths. Will the gain of the growth-based scheme still hold? Figure 7 shows that, under the new conditions, the gain is still substantial, although slightly smaller than the previous case with fixed initial length. This is because some customers require large initial lengths. When the space is relatively full, there may not be an empty location of sufficient size available to fit in the new customer. This will also cause fragmentation.

It is worth pointing out that the growth-based scheme reduces to the bisection scheme when the growth rates of all customers are much slower than the rate of increase in the number of customers. In this case, the growth-based scheme can be viewed as equivalent to the bisection scheme in the limit of zero growth rate.

5.2 Address Fragmentation

As the address space becomes more crowded, the probability of address fragmentation increases. The growth-based scheme can significantly reduce the amount of the fragmentation and delay the onset of fragmentation until a large percentage of the address space is utilized, as shown in Figure 8.

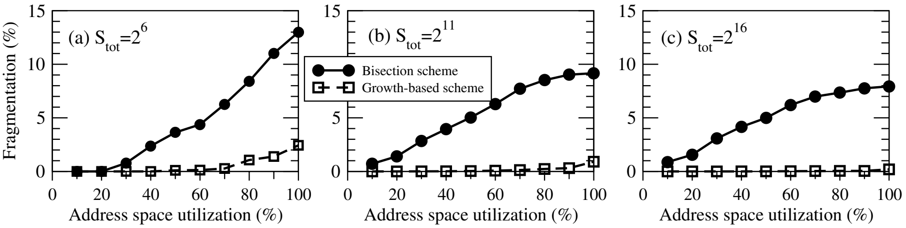


Fig. 8. Percentage of fragmented addresses for both bisection and growth-based schemes as a function of address space utilization for different sizes of total address space: (a) $S_{tot}=2^6$, (b) $S_{tot}=2^{11}$, and (c) $S_{tot}=2^{16}$

Using the bisection scheme, as the space utilization increases, the percentage of fragmented addresses increases steadily for all three sizes of total address space considered. Given the parameter set we used in this simulation, close to 8% of the addresses are fragmented when the utilization is over 70% given a total address space of 2^{11} (Figure 8(b)) or 2^{16} (Figure 8(c)). In addition, fragmentation takes place at small utilization percentages because of a few fast-growing customers.

In contrast, the fragmentation percentage remains to be small even at 100% space utilization for the growth-based scheme. This is especially true when the total address space is large, e.g., 2^{16}, giving the address provider enough time adjust for the fast-growing customers. Furthermore, the onset of fragmentation does not take place until ~40% of space utilization with the growth-based scheme. With this level of utilization, an ISP is qualified to obtain new address blocks [16]. Therefore, for the cases simulated, address fragmentation caused by allocation can be eliminated by using the growth-based scheme.

6 Conclusions

The current algorithm suggested by address allocation registries uses a bisection scheme for IPv6 address allocation. An improved version of this algorithm is proposed in this paper to find better address locations for the customers according to their growth-rates. Quantitative analysis of the performances of the two algorithms are conducted based on both theoretical models and simulations. Our studies show that the growth-based algorithm offers significant advantages over the bisection algorithm for both address aggregation and conservation. The benefits of the growth-based algorithm can be realized at all levels of the address allocation hierarchy.

Acknowledgments

We gratefully acknowledge Balaji Prabhakar, Pankaj Gupta, Larry Dunn, and Tony Hain for their valuable discussions and support.

References

1. S. Deering, R. Hinden. RFC2460 Internet Protocol, Version 6 (IPv6) Specification.
2. R. Hinden, S. Deering. RFC2373 IP Version 6 Addressing Architecture.
3. R. Hinden, S. Deering. RFC2374 IPv6 Aggregatable global unicast address format.
4. Z. Xu, X. Meng, C. J. Wittbrodt, S. Lu, L. Zhang. IPv4 Address Allocation and the Evolution of the BGP Routing Table. UCLA Computer Science Department, 2003.
5. H. Narayan, R. Govindan, G. Varghese. The Impact of Address Allocation and Routing on the Structure and Implementation of Routing Tables. SIGCOMM 2003.
6. http://6bone.net/
7. T. Bu, L. Gao and D. Towsley. On Characterizing Routing table growth. Proceedings of Global Internet 2002.
8. APNIC, RIPE, ARIN. IPv6 Address Allocation and Assignment Policy, June 2002.
9. G. Huston. Allocations vs. Anouncements, The ISP Column, May 2004.
10. IAB, IESG. RFC 3177 IAB/IESG Recommendations on IPv6 address Allocations to Sites, September 2001.
11. P. Wilson, R. Plzak, A. Pawlik. IPv6 Address Space Management, October 2002.
12. M. Kuhne, P. Rendek, S. Wilmot, L. Vegoda. IPv4 Address Allocation and Assignment Policies for the RIPE NCC Service Region, October 2003.
13. ARIN. IPv4 Policies. http://www.arin.net/policy/ipv4.html
14. Policies for IPv4 Address Space Management in Asia Pacific Region, March 2004.
15. K. Hubbard, M. Kosters, D. Conrad, K. Karrenberg, J. Postel. RFC 2050 Internet Registry IP Allocation Guidelines, November 1996.
16. ARIN. IPv6 Policies. http://www.arin.net/policy/ipv6_policy.html
17. IPv4 Address Space Report, http://bgp.potaroo.net/ipv4/
18. A. Durand. RFC 3194 The HD Ratio for Address Assignment Efficiency

Are Multiple Descriptions Better Than One?

György Dán, Viktória Fodor, and Gunnar Karlsson

Department of Signals, Sensors and Systems,
KTH, Royal Institute of Technology
{gyuri, viktoria, gk}@imit.kth.se

Abstract. In this paper we compare three schemes for error recovery for real-time multimedia applications: media-dependent forward error correction (MD-FEC), proposed for real-time audio; media-independent forward error correction (MI-FEC), proposed for real-time video; and multiple description coding (MDC). We provide a detailed queueing analysis for these schemes considering bursty traffic sources, and combine results from information theory and queueing theory to analyze their performance. We conclude that MDC always performs better than MD-FEC, and that the average loss probability plays a key role in the choice of the optimal parameters. We also show that MDC outperforms MI-FEC if packet losses are highly correlated, like in the current Internet, and if the delay available for error control is low.

1 Introduction

Applications that require low loss probabilities in today's Internet have to employ some end-to-end mechanism to recover from packet losses. For interactive applications with strict delay constraints, the delay introduced by the applied schemes has to be low as well. Forward error correction (FEC) and the recently re-discovered multiple description coding (MDC) have been proposed for this purpose [1, 2, 3].

There are two main directions of FEC design to recover from packet losses. One solution, proposed by the IETF and implemented in Internet audio tools like Rat, is to add a redundant copy of the original packet to one of the subsequent packets. The redundant packet is highly compressed, so that the sound quality reconstructed from the redundant packet is low, but still better than when there is nothing to play out. Proposed ways to improve the performance of the scheme are to increase the time offset between the transmission of the original packet and the redundant one [4], and to send multiple redundant copies in subsequent packets [1]. The performance of this media-dependent FEC scheme (MD-FEC) has been evaluated via simulations in [5] and analytically in [6]. The results show that MD-FEC can be beneficial if the share of the total traffic implementing it is small.

R. Boutaba, J.P. Black (Eds.): NETWORKING 2005, LNCS 3462, pp. 684–696, 2005.

The other set of FEC solutions use block coding schemes based on algebraic coding, e.g. Reed-Solomon coding. The main idea behind these media-independent FEC schemes (MI-FEC) is to protect a block of k packets of information with c packets of redundancy. Thus the original k packets can be reconstructed if out of $n = k + c$ packets at most c packets are lost. MI-FEC has mainly been proposed for low-delay high-bitrate multimedia applications, such as real-time video, and has been extensively studied both via simulations and analytically [2, 7]. The results are similar to those of MD-FEC: MI-FEC is efficient if only a small share of the streams uses it and the efficiency of MI-FEC increases as the block length, and thus the introduced delay, increases.

The traditional approach to networking, the separation of source and channel coding, was motivated by Shannon's separation theorem [8]. It says that source coding and channel coding can be performed separately while maintaining optimality. However, Shannon's separation theorem assumes that the available delay is unlimited, which is not true for real-time applications. MDC addresses the problem of joint source and channel coding. Originally it was designed for the transmission of multiple descriptions of a single source over independent channels. If only one of the descriptions is received, it is used for reconstruction with a certain accuracy. If more than one descriptions are received, then the information from the other descriptions can be used to enhance the accuracy. It has been rediscovered recently for use in packet switched networks [3]: instead of using separate channels, one can put the different encodings into different packets and send them with a time offset, similarly to the case of MD-FEC. In general the amount of information sent over the separate channels can be different, however in single-path packet networks it can be shown that balanced MDC, i.e. the one sending the same amount of information in all packets is optimal [3].

In this paper we compare the performance of the two FEC schemes and MDC via combining results from information theory and queueing theory. The optimal selection of source and redundancy rates for real-time audio was investigated before in [1] using the Gilbert channel model. A similar approach was followed in [9] with respect to MPEG coded video. The performance of MDC was compared to that of single description coding in the context of content delivery networks via simulations in [10]. The authors considered an average packet loss rate of 5% and concluded that MDC can reduce the distortion by as much as 20 to 40%. The performance of MI-FEC and MDC with 50% redundancy was compared in [11] using the Gilbert channel model. As the authors used a fixed redundancy rate for FEC, the comparison gave advantage to MDC by allowing it to adjust to the channel characteristics.

The rest of the paper is organized as follows. In Section 2 we describe the queueing model used to calculate the conditional loss probability of packets. In Section 3 we introduce the distortion rate bounds for FEC and MDC. In Section 4 we compare the performance of MD-FEC and MDC to be used for low-bitrate multimedia streams. In Section 5 we compare the performance of MI-FEC and MDC suitable for high-bitrate multimedia streams. We conclude our work in Section 6.

2 Analysis of the Packet Loss Process

In this section we first give an overview of past work on the modeling of the packet loss correlation, then we turn to the calculation of the conditional packet loss probability. We will use the conditional loss probability and the probability of j losses in a block of n packets in the remainder of the paper to calculate the average distortion of the various error correction schemes under different network conditions.

Schulzrinne et al. derived the consecutive loss probability for the $N * IPP/D/1/K$ queue and showed that it can be orders of magnitude higher than the loss probability [12]. The conditional loss probability was analyzed under the assumption of exponential interarrival time and service time distribution in [6]. The authors considered a single flow and a packet spacing smaller than the queue length. They also presented a model with exponential interarrival time and general service time distribution and gave simple lower and upper bounds on the consecutive loss probability.

2.1 Model Description

We model the network as a single queue, representing the bottleneck in the transmission path [13, 14], with Erlang-r distributed packet sizes having average transmission time $1/\mu$. Using the Erlang-r distribution makes the mathematical analysis simple, while it can be used to get approximate results on the performance of FEC by matching the mean and variance of the packet size distribution to measurements [15]. Packets arrive to the system from two sources, a Markov-modulated Poisson process (MMPP) and a Poisson process, representing the tagged source and the background traffic respectively. The packets are stored in a buffer that can host up to K packets, and are served according to a FIFO policy.

It is known that compressed media, e.g. VBR video, exhibits a self-similar nature [16]. Ryu and Elwalid [17] showed, however, that short term correlations have dominant influence on the network performance under realistic scenarios of buffer sizes for real-time traffic. Thus the MMPP may be a practical model to derive approximate results for the queueing behavior of compressed media, especially in the case of small buffer sizes.

The assumption of the Poisson background traffic is partly justified by recent results indicating that Internet traffic can be approximated by a non-stationary Poisson process [18]. According to measurements, the change free intervals are well above 150 ms, the ITU's G.114 recommendation for end-to-end delay for real-time applications. These empirical results are consistent with recent theoretical results [19].

We assume that the sources feeding the system are independent. The MMPP is described by the infinitesimal generator matrix Q with elements r_{lm} and the arrival rate matrix $\Lambda = diag\{\lambda_1, \ldots, \lambda_L\}$, where λ_l is the average arrival rate while the underlying Markov chain is in state l [20]. The Poisson process modeling the background traffic has average arrival rate λ. The superposition of the two sources can be described by a single MMPP with arrival rate matrix $\hat{\Lambda} = \Lambda \oplus \lambda = \Lambda + \lambda I$, and infinitesimal generator $\hat{Q} = Q$, where $\oplus$ is the Kronecker sum. Each packet in the queue corresponds to r exponential stages of service, and the state space of the queue is $\{0, \dots, rK\} \times \{1, \dots, L\}$.

2.2 Conditional Packet Loss

Our purpose is to calculate the probability of the event that an arbitrary packet from the tagged source arrives in state i of the system and the n^{th} next packet arrives in state j of the system $P_{i,j}(n)$, $n \geq 0$, $0 \leq i, j \leq rK$. We define the probability $P_{j|i}^{m|l}(n)$ (and $\overline{P}_{j|i}^{m|l}(n)$), $0 \leq i, j \leq rK$, $1 \leq l, m \leq L$, $n \geq 0$ ($n \geq 1$) as the probability of the event that the n^{th} next packet from the tagged source is generated while the MMPP is in state m and arrives in state j of the system, given that the remaining number of exponential stages in the system is i just before the arrival of the first (next) packet from the tagged stream (background traffic) and the MMPP is in state l. As the first packet is arbitrary,

$$P_{i,j}(n) = \sum_{l=1}^{L} \Pi(i,l) \sum_{m=1}^{L} P_{j|i}^{m|l}(n). \tag{1}$$

$\Pi(i,l)$, the steady state distribution of the exponential stages in the queue as seen by an arriving packet can be calculated as

$$\Pi(i,l) = \frac{\pi(i,l)\lambda_l}{\sum_{l=1}^{L} \lambda_l \sum_{i=0}^{rK} \pi(i,l)}, \tag{2}$$

where $\pi(i,l)$ is the steady state distribution of the $MMPP/E_r/1/K$ queue [15].

The probabilities $P_{j|i}^{m|l}(n)$ can be derived according to the following recursion. The recursion is initiated for $n = 0$ with the following relations

$$P_{j|i}^{m|l}(0) = \begin{cases} 1 & \text{if} \quad j = i \text{ and } m = l \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Using the notation $p_u = \frac{\lambda_u}{\lambda_u + \lambda}$ and $\overline{p}_u = \frac{\lambda}{\lambda_u + \lambda}$, for $n \geq 1$ the following equations hold.

$$P_{j|i}^{m|l}(n) = \sum_{u=1}^{L} \sum_{k=0}^{i+r} Q_{i+r}^{u|l}(k) \{ p_u P_{j|i+r-k}^{m|u}(n-1) + \overline{p}_u \overline{P}_{j|i+r-k}^{m|u}(n-1) \} \quad 0 \leq i \leq r(K-1), \tag{4}$$

$$P_{j|i}^{m|l}(n) = \sum_{u=1}^{L} \sum_{k=0}^{i} Q_{i}^{u|l}(k) \{ p_u P_{j|i-k}^{m|u}(n-1) + \overline{p}_u \overline{P}_{j|i-k}^{m|u}(n-1) \} \quad r(K-1) < i, \tag{5}$$

$$\overline{P}_{j|i}^{m|l}(n) = \sum_{u=1}^{L} \sum_{k=0}^{i+r} Q_{i+r}^{u|l}(k) \{ p_u P_{j|i+r-k}^{m|u}(n) + \overline{p}_u \overline{P}_{j|i+r-k}^{m|u}(n) \} \quad 0 \leq i \leq r(K-1), \tag{6}$$

$$\overline{P}_{j|i}^{m|l}(n) = \sum_{u=1}^{L} \sum_{k=0}^{i} Q_{i}^{u|l}(k) \{ p_u P_{j|i-k}^{m|u}(n) + \overline{p}_u \overline{P}_{j|i-k}^{m|u}(n) \} \quad r(K-1) < i. \tag{7}$$

$Q_i^{u|l}(k)$ denotes the joint probability that the next arrival will be in state u of the MMPP and that k exponential stages out of i will be completed before the next arrival from the joint arrival process, given that the last arrival was in state l of the MMPP. A way to calculate $Q_i^{u|l}(k)$ is shown in [15]. The procedure of computing $P_{j|i}^{m|l}(n)$ follows that of computing $P(j,n)$ in [15].

Let us now define two sets of states, α and ω as the set of states of the queue where arriving packets can enter the system and where arriving packets are discarded respectively. Then $\alpha = \{0 \dots r(K-1)\}$ and $\omega = \{r(K-1)+1 \dots rK\}$. Using these notations the probability that a packet arriving to the system from the tagged source is not lost and the n^{th} next packet from the tagged source is lost is given as $p_{\alpha\omega}(n) = \sum_{i\in\alpha}\sum_{j\in\omega} P_{i,j}(n)$. The probabilities $p_{\alpha\alpha}(n)$, $p_{\omega\alpha}(n)$ and $p_{\omega\omega}(n)$ can be defined in a similar way. Throughout the paper we refer to $p_{\omega|\omega}(n)$ as the consecutive loss probability for $n = 1$ and as the conditional loss probability for $n \geq 1$.

2.3 Losses in a Block

We use the method presented in [15] to calculate the probability of j losses in a block of n packets $P(j,n)$, $n \geq 1$, $0 \leq j \leq n$ for the described network model. Based on the probabilities $P(j,n)$, one can calculate the probability that a packet is uncorrectable by MI-FEC, given the parameters n and k, as $P_{uc}(n,k) = \sum_{j=n-k+1}^{n} jP(j,n)/n$.

3 Distortion-Rate Bounds for FEC and MDC

When compressing signals, there is always a tradeoff between the size and the accuracy of the representation. This tradeoff, the bounds on achievable rates and distortions, is represented by the distortion-rate and the rate-distortion functions, which depend on the source characteristics and the distortion measure. We consider a memoryless Gaussian source with unit variance and use the mean squared error distortion measure, which is the most common distortion measure. The distortion-rate function for a Gaussian source with unit variance and the squared distortion measure is given as

$$D(R) = 2^{-2R}, \tag{8}$$

where R is the code rate and $D(R)$ is the distortion [3]. In the following we discuss the distortion-rate characteristics of the three considered error control schemes.

In the case of MD-FEC the first and the subsequent $\nu - 1$ (redundant) descriptions are encoded independently of each other, and thus, if any of them is received, its distortion is given by (8). We denote the rate allocated to the primary encoding with R_1^{MDF} and the rate allocated to the k^{th} (redundant) copy with R_k^{MDF}. The redundancy ratio introduced by the MD-FEC is then $\beta = \sum_{k=2}^{\nu} R_k^{MDF}/R_1^{MDF}$, and the total rate of the source is $R^{MDF} = R_1^{MDF}(1+\beta)$. If both the primary and some redundant encodings are received, the redundant encodings can not be used to reduce the distortion.

In the case of MI-FEC each packet corresponds to an encoding of rate R_1^{MIF}. The packets compose blocks, and redundancy is added to the block of packets. If we denote the ratio of redundancy with $\beta = (n-k)/k$, then the total rate is given as $R^{MIF} = R_1^{MIF}(1+\beta)$. If the number of lost packets in a block is no more than c then the original packets can be reconstructed and the distortion of the individual packets is given by (8).

In the case of MDC we denote the total source rate with R^{MDC} and consider the balanced two channel case. We denote the rate allocated to individual descriptions with $R_1^{MDC} = R^{MDC}/2$. The distortion when both descriptions are received, called the central distortion, is denoted by D_0^{MDC} and the distortion if only one of the descriptions is received, called the side distortion, is denoted with D_1^{MDC}. The distortion rate bounds for the 2-channel MDC are [3]:

$$D_1^{MDC} \geq 2^{-2R^{MDC}/2} \tag{9}$$

$$D_0^{MDC}(D_1^{MDC}) \geq 2^{-2R^{MDC}} \gamma(R^{MDC}, D_1^{MDC}), \tag{10}$$

where $\gamma(R^{MDC}, D_1^{MDC}) = 1$ if $2D_1^{MDC} > 1 + D_0^{MDC}$ and otherwise

$$\gamma(R^{MDC}, D_1^{MDC}) = \{1 - \{(1 - D_1^{MDC}) - \sqrt{(D_1^{MDC})^2 - 2^{-2R^{MDC}}}\}^2\}^{-1}. \tag{11}$$

Equation (10) shows that if the side distortion is small then the central distortion is higher than the distortion rate minimum (8). For a primer on rate-distortion theory and multiple description coding see [3].

In general the source rate R (bits per symbol) and the bitrate of the stream C (bits per second) are related to each other through the symbol rate S (symbols per second) as $C = S \times R$. For example, a video sequence with a small image size can reach a higher source rate R at the same bitrate as another sequence with a big image size. For this reason in the following analysis we will distinguish between these two parameters.

4 MD-FEC Versus MDC

In this section we consider a source that has an available source rate R_a and can use either MD-FEC or MDC for error control due to the low delay bounds and low bitrate. For simplicity, we will consider the case of $\nu = 2$. Using the notations introduced in Section 3 the mean distortion bound of the MD-FEC scheme can be calculated as the weighted sum of the distortions of the cases when both descriptions are received, when only one of them is received or when none of them is received

$$D^{MDF}(\beta) = (p_{\alpha\alpha}(n) + p_{\alpha\omega}(n))D(R_1^{MDF}) + p_{\omega\alpha}(n)D(R_2^{MDF}) + p_{\omega\omega}(n), \tag{12}$$

where the loss probabilities were defined in Section 2.2. For a redundancy ratio β the rates of the individual descriptions are $R_1^{MDF} = R_a/(1+\beta)$ and $R_2^{MDF} = R_a\beta/(1+\beta)$.The level of redundancy that minimizes (12) is the solution β_*^{MDF}

of $\frac{\partial D^{MDF}(\beta)}{\partial \beta} = 0$. Using the distortion rate function introduced in Section 3 the solution is

$$\beta_*^{MDF} = \frac{ln(2^{2R_a}) - ln(\frac{p_{\alpha\alpha}(n)+p_{\omega\alpha}(n)}{p_{\omega\alpha}(n)})}{ln(2^{2R_a}) + ln(\frac{p_{\alpha\alpha}(n)+p_{\omega\alpha}(n)}{p_{\omega\alpha}(n)})} = \frac{ln(2^{2R_a}) - ln(\frac{1-p_\omega(n)}{p_\omega(n)(1-p_{\omega|\omega}(n))})}{ln(2^{2R_a}) + ln(\frac{1-p_\omega(n)}{p_\omega(n)(1-p_{\omega|\omega}(n))})}. \tag{13}$$

We denote the minimal mean distortion by D_*^{MDF}. Based on (13) the optimal ratio of redundancy β_*^{MDC} is positive only if

$$p_\omega > \{1 + D(R_a)^{-1}(1 - p_{\omega|\omega}(n))\}^{-1} \tag{14}$$

Thus for a given source rate it is not beneficial to use MD-FEC below this average loss probability. Furthermore, if $p_{\omega|\omega}(n)$ is not close to 1, the minimum value of p_ω is dominated by $D(R_a)^{-1}$. In general, for any distortion rate function of the form $D(R) = ab^{-cR}$ the limit given in (14) is $p_\omega > \{1 + aD(R_a)^{-1}(1 - p_{\omega|\omega}(n))\}^{-1}$.

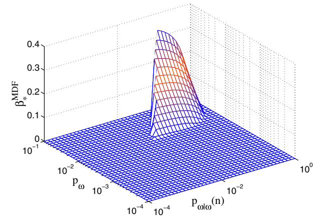


Fig. 1. Optimal rate of redundancy vs. p_ω and $p_{\omega|\omega}(n)$ for $D(R_a) = 0.01$

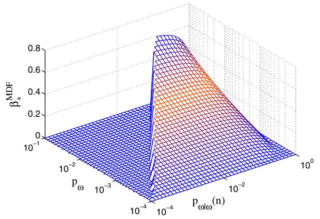


Fig. 2. Optimal rate of redundancy vs. p_ω and $p_{\omega|\omega}(n)$ for $D(R_a) = 10^{-4}$

By substituting β_*^{MDF} given by (13) into (12) it can be shown analytically that increasing n decreases the average distortion whenever p_ω satisfies (14) since $p_{\omega|\omega}(n)$ is a decreasing function of n. This result is in accordance with the empirical observations presented in [4]. If, however, $p_\omega \leq 1/(1 + D(R_a)^{-1})$ then increasing n does not decrease the distortion. The value β_*^{MDF} that minimizes the average distortion at a given average loss probability and conditional loss probability is shown in Fig. 1 and 2 for distortions $D(R_a) = 10^{-2}$ and $D(R_a) = 10^{-4}$ respectively. Comparing the figures we see that the rate of the source and the average loss probability have a big influence on whether or not MD-FEC should be used.

Now we consider the case of the balanced MDC scheme with two descriptions. The total available rate is R_a, and thus the rates of the individual descriptions are $R_1^{MDC} = R_2^{MDC} = R_a/2$. The mean distortion bound can be calculated as

$$D^{MDC}(D_1^{MDC}) = p_{\alpha\alpha}(n)D_0^{MDC}(D_1^{MDC}) + (p_{\alpha\omega}(n) + p_{\omega\alpha}(n))D_1^{MDC} + p_{\omega\omega}(n). \tag{15}$$

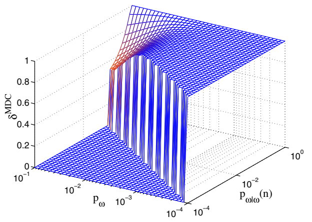


Fig. 3. MDC gain vs. p_ω and $p_{\omega|\omega}(n)$ for $D(R_a) = 0.01$

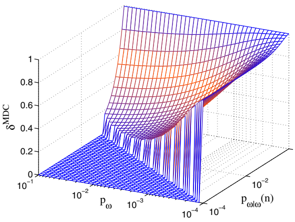


Fig. 4. MDC gain vs. p_ω and $p_{\omega|\omega}(n)$ for $D(R_a) = 10^{-4}$

The optimal combination of central and side distortions can be found by setting the first derivative with regard to D_1^{MDC} of (15) to zero [11]. The solution is given by

$$\frac{\partial D_0^{MDC}(D_1^{MDC})}{\partial D_1^{MDC}} = -\frac{p_{\alpha\omega}(n) + p_{\omega\alpha}(n)}{p_{\alpha\alpha}(n)} = -\frac{p_\omega(1 - p_{\omega|\omega}(n))}{1 + p_{\omega|\omega}(n)p_\omega - 2p_\omega}. \tag{16}$$

We denote the minimal mean distortion by D_*^{MDC}. To compare the performance of MD-FEC and MDC we define $\delta_{MDF}^{MDC} = D_*^{MDC}/D_*^{MDF}$, the ratio of the mean distortion bounds. Similarly we define $\delta^{MDC} = D_*^{MDC}/(p_\alpha D(R_a)+p_\omega)$, the ratio of the mean distortion bound with MDC to the bound without any error control. Figs. 3 and 4 show δ^{MDC} as a function of the average loss probability and the conditional loss probability for distortions $D(R_a) = 10^{-2}$ and $D(R_a) = 10^{-4}$. They show that the available source rate, the average loss probability and the conditional loss probability together determine the potential of MDC to decrease the mean distortion. Figs. 5 and 6 show δ_{MDF}^{MDC} as a function of the average loss probability and the conditional loss probability for distortions $D(R_a) = 10^{-2}$ and $D(R_a) = 10^{-4}$. They show that MDC reduces the mean distortion more efficiently than MD-FEC under all circumstances. The difference, however, is not large, up to a factor of 25%. Comparing these figures to Figs. 1 and 2, we see that MDC outperforms MD-FEC primarily in those regions of average loss and conditional loss probability where $\beta_*^{MDF} > 0$. It follows that error correction with MDC becomes practically useless below the loss probability given in (14).

The value of the parameters p_ω and $p_{\omega|\omega}(n)$ used in the evaluation presented above can be taken from measured traces, simulations or mathematical models. In the remainder of the section we use the model presented in Section 2 to evaluate the effects of various traffic and network parameters on the loss correlation. We consider a scenario, possibly VoIP transmission, where the average arrival intensity is $50/sec$ and the packets have a constant size of 160 bytes, thus the corresponding bitrate would be 64 kbps. The interarrival time between packets from the tagged source is exponentially distributed. This assumption can be justified by the fact that packets from a flow cross several routers before arriving to the bottleneck router [6]. The background traffic is modeled by a Poisson process, packet sizes are exponentially distributed with an average packet length of 454

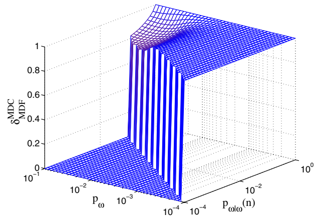


Fig. 5. δ_{MDF}^{MDC} vs. p_ω and $p_{\omega|\omega}(n)$ for $D(R_a) = 0.01$

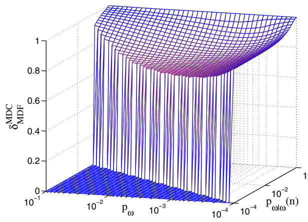


Fig. 6. δ_{MDF}^{MDC} vs. p_ω and $p_{\omega|\omega}(n)$ for $D(R_a) = 10^{-4}$

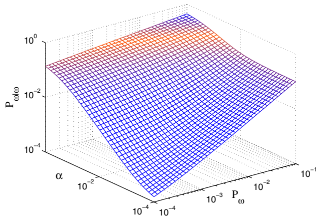


Fig. 7. $p_{\omega|\omega}(1)$ vs. average loss probability and α for K=5

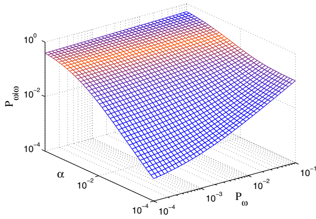


Fig. 8. $p_{\omega|\omega}(1)$ vs. average loss probability and α for K=20

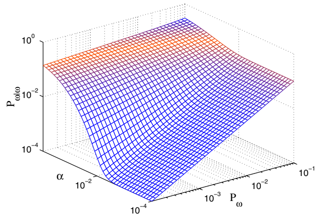


Fig. 9. $p_{\omega|\omega}(4)$ vs. average loss probability and α for K=20

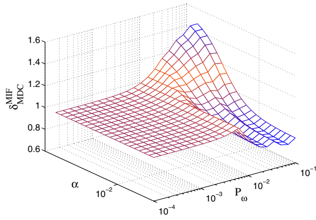


Fig. 10. δ_{MDC}^{MIF} vs. average loss probability and α for $K = 5$, $n = 10$ and $D(R_a) = 10^{-2}$

bytes. This value closely matches the average packet size of the Internet traffic traces shown in [21]. We denote by α the probability that an arriving packet belongs to the tagged source, and consider values between 10^{-4} and 1. These values correspond to a background traffic between 0 and 226 Mbps. Depending on the value of α the packet size distribution of the aggregate traffic will differ from exponential. We approximate the resulting packet size distribution by means of the Erlang-r distribution. For a given value of α we change the link capacity to achieve different average loss probabilities, between 10^{-4} and 10^{-1}. Figs. 7 and 8 show $p_{\omega|\omega}(1)$ as a function of the average loss probability and α for a queue capacity of $K = 5$ and $K = 20$ respectively. The figures show that at high loss rates the conditional loss probability reaches the average loss probability faster than at low loss rates. Fig. 9 shows $p_{\omega|\omega}(4)$ (a spacing of 4 packets corresponds to a delay of 80 ms) as a function of p_ω and α for queue capacity $K = 20$. The figure shows that the lower the value of α the higher the decrease of $p_{\omega|\omega}$ due to the increased delay. This has been observed in measurements on the public Internet [22].

5 MI-FEC Versus MDC

In the following section we consider an arbitrary packet, part of a high bitrate multimedia stream and evaluate how MI-FEC and MDC can protect the data against losses. Using the notations of Section 3 the mean distortion bound of MI-FEC for given block length n, ratio of redundancy β and available rate R_a is

$$D^{MIF}(n,k) = (1 - P_{uc}(n,k))D(\frac{R_a}{1+\beta}) + P_{uc}(n,k), \tag{17}$$

where $P_{uc}(n,k)$ was defined in Section 2.3. We consider the case that the block length n is given, and if the source wants to increase the ratio of redundancy then it has to decrease its source rate. Thus in order to minimize the mean distortion we select the value of k that minimizes (17), and denote the minimum by D_*^{MIF}.

The mean distortion bound of the two-channel MDC and the optimal combination of side and central distortions for a given loss process have been shown in (15) and (16) in Section 4. We choose the spacing between the two descriptions to be $n-1$ packets, where n is the block length of MI-FEC. This way the same delay is introduced by the two schemes.

In the following we consider a scenario, where the tagged source is a 3-state Markov modulated Poisson process with arrival intensities $\lambda_1 = 116/s, \lambda_2 = 274/s, \lambda_3 = 931/s$ and transition rates $r_{12} = 0.12594, r_{21} = 0.25, r_{23} = 1.97, r_{32} = 2$. These values were derived from an MPEG-4 coded trace by matching the average bitrates of the I, P and B frames and the corresponding transition intensities. We use the same model for the MD coded video as there are no models proposed for it. The packet size of the tagged source is fixed to 188 bytes, as for the transport stream in the MPEG-2 standard [23]. Based on these parameters the bitrate of the source is 540 kbps. The assumptions on the background traffic, the aggregate packet size distribution and the definition of α are the same as in Section 4. We change the value of α between 10^{-3} and 1, these values correspond to a background traffic of 0 to 162 Mbps. We change the link capacity to achieve different average loss probabilities between 10^{-4} and 10^{-1}.

To compare the performance of MI-FEC and MDC we define $\delta_{MDC}^{MIF} = D_*^{MIF}/D_*^{MDC}$, the ratio of the mean distortion bounds. Fig. 10 shows δ_{MDC}^{MIF} as a function of the average loss probability and α for a source with distortion $D(R_a) = 10^{-2}$, buffer capacity $K = 5$ and block length $n = 10$ ($\approx$ 30 ms delay). The figure shows that below the loss probability given in (14) neither MI-FEC nor MDC can decrease the distortion. Above this loss probability MDC outperforms MI-FEC if α is high, e.g. losses are correlated (see Fig. 7). The reason for this phenomenon is that the redundant information needed to recover a lost packet with MI-FEC is carried in subsequent packets. Thus if losses occur in bursts, a high ratio of redundancy is needed to recover from losses. In the case of MDC however, descriptions are placed as far away as possible and so the loss process as seen by MDC is less bursty. Measurements performed on the Internet [22] confirm that losses occur independently above spacing $n > 20$. The “valley” shapes parallel to the α axis are due to the fact that β for MI-FEC can not be adjusted continuously. Figure 11 shows the same scenario for a block length of $n = 20$ ($\approx$ 60 ms delay). Comparing the figures shows an increasing performance

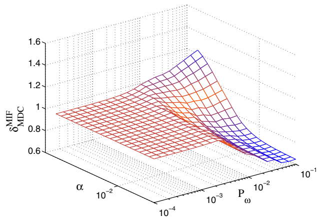


Fig. 11. δ^{MIF}_{MDC} vs. average loss probability and α for $K = 5$, $n = 20$ and $D(R_a) = 10^{-2}$

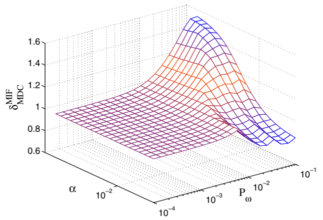


Fig. 12. δ^{MIF}_{MDC} vs. average loss probability and α for $K = 20$, $n = 10$ and $D(R_a) = 10^{-2}$

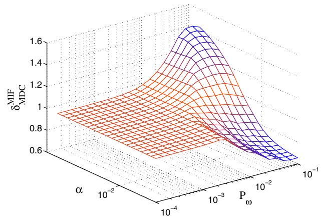


Fig. 13. δ^{MIF}_{MDC} vs. average loss probability and α for $K = 20$, $n = 20$ and $D(R_a) = 10^{-2}$

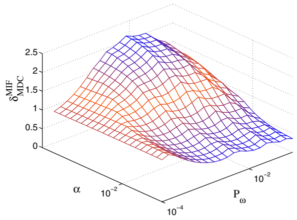


Fig. 14. δ^{MIF}_{MDC} vs. average loss probability and α for $K = 20$, $n = 10$ and $D(R_a) = 10^{-4}$

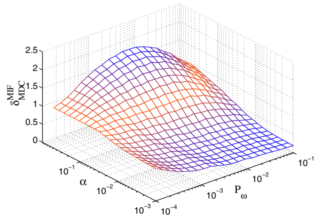


Fig. 15. δ^{MIF}_{MDC} vs. average loss probability and α for $K = 20$, $n = 20$ and $D(R_a) = 10^{-4}$

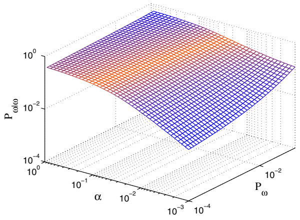


Fig. 16. $p_{\omega|\omega}(1)$ vs. average loss probability and α for K=20, bursty traffic

of MI-FEC as the block length increases. In fact, as the block length increases, MI-FEC can correct longer bursts at the same redundancy ratio. By increasing the buffer size, M to 20, we see that MDC performs better over a wider range of parameters p_ω and α as shown in Figs. 12 and 13. This is due to the higher correlation between losses as an effect of the increased queue length (see Figs. 7 and 8). Figs. 14-15 show results for the same scenarios for distortion $D(R_a) = 10^{-4}$.

Recent measurements on the Internet [22, 24] show that the average loss probability is around 1% while the consecutive loss probability around 25%. Considering $K = 20$ and $D(R_a) = 10^{-4}$, these parameters correspond to $\alpha = 0.0305$ (Fig. 16), and thus for $n = 5, 10$ and 20 we get $\delta^{MIF}_{MDC} = 2.32, 1.5$ and 0.91 respectively (Figs. 14 and 15). The choice of an average loss probability of 5% and consecutive loss probability 20% as in [10] corresponds to $\alpha = 0.016$ and leads to $\delta^{MIF}_{MDC} = 1.45, 0.96$ and 0.59 for $n = 5, 10$ and 20 respectively. These examples suggest that for short delays MDC performs better than MI-FEC on today's Internet.

6 Conclusions and Discussion

In this paper we presented an analytical evaluation of the potential of forward error correction and multiple description coding to recover from losses under various network conditions. We compared media-dependent FEC to MDC and concluded that MDC gives better performance under all circumstances. The comparison is based on the loss probabilities, thus it is independent of the traffic parameters. We showed that for a big class of distortion-rate models, the optimal ratio of redundancy is mainly determined by the average packet loss probability. We proposed a queueing model to evaluate the loss process of bursty traffic and used the model to compare the performance of media independent FEC and MDC with two descriptions. We showed that MDC gives better performance if losses are correlated and the delay available for error control is low. In today's Internet losses are correlated and network end-to-end delays are large, thus leaving short delays for error control. Hence we conclude that even MD coding with two descriptions could improve the quality of real-time multimedia communications.

Based on the results presented in this paper we believe that multiple description coding is an appealing error control solution for delay sensitive traffic in an environment with correlated packet losses like the Internet.

References

1. J. C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for Internet telephony," in *Proc. of IEEE INFOCOM*, pp. 1453–1460, 1999.
2. E. Biersack, "Performance evaluation of forward error correction in ATM networks," in *Proc. of ACM SIGCOMM*, pp. 248–257, August 1992.
3. V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Processing Mag.*, September 2001.

4. I. Kouvelas, O. Hodson, W. Hardman, and J. Crowcroft, "Redundancy control in real-time Internet audio conferencing," in *Proc. of International Workshop on Audio-Visual Services Over Packet Networks*, September 1997.
5. M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-based error control for packet audio on the Internet," in *Proc. of IEEE INFOCOM*, pp. 505–515, 1998.
6. E. Altman, C. Barakat, and V. Ramos, "Queuing analysis of simple FEC schemes over IP," *Computer Networks*, vol. 39, pp. 185–206, June 2002.
7. K. Kawahara, K. Kumazoe, T. Takine, and Y. Oie, "Forward error correction in ATM networks: An analysis of cell loss distribution in a block," in *Proc. of IEEE INFOCOM*, pp. 1150–1159, June 1994.
8. C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, pp. 379–423, 1948.
9. P. Frossard and O. Verscheure, "Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery," *IEEE Trans. Image Processing*, vol. 10, no. 12, pp. 1815–1825, 2001.
10. J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proc. of IEEE INFOCOM*, pp. 1736–1745, 2002.
11. M. Y. Kim and B. W. Kleijn, "Rate-distortion comparisons between FEC and MDC based on Gilbert channel model," in *Proc. of IEEE ICON*, pp. 495–500, 2003.
12. H. Schulzrinne, J. Kurose, and D. Towsley, "Loss correlation for queues with bursty input streams," in *Proc. of IEEE ICC*, pp. 219–224, 1992.
13. J. C. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *Proc. of ACM SIGCOMM*, pp. 289–298, September 1993.
14. O. J. Boxma, "Sojourn times in cyclic queues - the influence of the slowest server," *Computer Performance and Reliability*, pp. 13–24, 1988.
15. G. Dán, V. Fodor, and G. Karlsson, "Packet size distribution: an aside?," in *Proc. of QoS-IP'05*, pp. 75–87, February 2005.
16. J. Beran, R. Sherman, M. Taqqu, and W. Willinger, "Long-range dependence in variable-bit-rate video traffic," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 1566–1579, 1995.
17. B. Ryu and A. Elwalid, "The importance of long-range dependence of VBR video traffic in ATM traffic engineering: Myths and realities," in *Proc. of ACM SIGCOMM*, pp. 3–14, 1996.
18. T. Karagiannis, M. Molle, and M. Faloutsos, "A nonstationary poisson view of Internet traffic," in *Proc. of IEEE INFOCOM*, pp. 1–12, March 2004.
19. R. Gaigalas and I. Kaj, "Convergence of scaled renewal processes and a packet arrival model," *Journal of the Bernoulli Society for Mathematical Statistics and Probability*, vol. 9, no. 4, pp. 671–703, 2003.
20. W. Fischer and K. Meier-Hellstern, "The markov-modulated poisson process MMPP cookbook," *Performance Evaluation*, vol. 18, no. 2, pp. 149–171, 1992.
21. Sprint IP Monitoring Project, "http://ipmon.sprint.com/."
22. M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Meaurement and modelling of the temporal dependence in packet loss," in *IEEE INFOCOM*, pp. 345–352, 1999.
23. D. Hoffman and G. Fernando, "RTP payload format for MPEG1/MPEG2 video," RFC 2250, Jan. 1998.
24. I. Marsh, F. Li, and G. Karlsson, "Wide area measurements of VoIP quality," in *Proc. of Quality of Future Internet Services*, pp. 63–72, October 2003.

An Approximate Queueing Model for Limited-Range Wavelength Conversion in an OBS Switch

Vishwas S Puttasubbappa and Harry G Perros

North Carolina State University,
Computer Science Department, Raleigh NC 27695-7534
{vsputtas, hp}@csc.ncsu.edu

Abstract. We develop an analytical model for modeling limited-range wavelength conversion employed in an OBS switch. The system is modeled as a simultaneous resource possession problem. We propose a product-form solution which calculates approximate blocking probabilities for degree of conversion $d = 1, 2$ and for large number of wavelengths. We then propose an approximate model for large values of d. The output of our model was verified with simulation results.

1 Introduction

Optical Burst Switching (OBS) is one of the promising technologies to realize the next generation all-optical Internet [3] [8] [15]. In OBS, the contention issue that arises at the switches due to simultaneous arrival of several control packets can be minimized by using *wavelength conversion*, *Fiber Delay Line (FDL) buffers* and *deflection routing*. A combination of these three techniques can also be employed to reduce blocking of data bursts. C. Gauger [5] examined the performance of OBS nodes which employ wavelength converter pools and FDL buffers for contention resolution. Several strategies based on different ordering of probing converter pools and FDL buffers have been proposed to optimize performance such as minimizing delay or the number of converters.

In this paper, we focus on wavelength conversion to reduce contention among data bursts. We consider only the case of limited-range wavelength conversion. In limited-range wavelength conversion, data bursts arriving on a wavelength can be converted to a fixed set of wavelengths above and below the original wavelength. The degree of conversion d defines the number of target wavelengths for conversion on either side of the original wavelength. Thus an incoming data burst can be converted to a total of $(2 * d + 1)$ destination wavelengths. Previous studies on limited-range wavelength conversion for lightpaths include [4],[12] and [14] for an OBS network. Recently, Akar and Karasan [2] proposed a method to exactly calculate the blocking probabilities in an OBS switch with partial wavelength conversion, i.e number of converters available being less than the number of wavelengths.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 697–708, 2005.

Previous studies in OBS which dealt with limited-range wavelength conversion focused on computing link blocking probabilities and path blocking probabilities [14]. They also assumed Poisson arrivals for their model. In this paper, we focus on a single OBS switch. We assume sources to be of ON-OFF nature.

We develop an analytical model to determine the blocking probabilities for data bursts in a core OBS switch which employs limited-range wavelength conversion to resolve contention among data bursts. We assume the absence of FDLs and deflection routing. The blocking probabilities obtained from this queueing model are approximate. We propose a product-form solution from which blocking probabilities can be computed for large number of wavelengths but only for $d = 1, 2$. We then develop a large scale approximation technique which can be applied to large values of w and d.

The paper is organized as follows: Section 2 describes the architecture of the core OBS switch that we have assumed for our model. In Section 3, we present an approximate queueing model for limited-range wavelength conversion for the case of small d. In Section 4, we describe an approximate model for large systems. In Section 5, we compare the outputs of our analytical models with the simulation results, and in Section 6, we draw some conclusions.

2 The OBS Switch

The core OBS switch is comprised of k incoming and k outgoing fibers, the switching fabric and wavelength converters. Such a switch can be implemented using several architectures [13] [10] [11]. Each output fiber has a set of wavelength converters (c) that can be used by bursts traveling out of that particular fiber. The bandwidth in each fiber is partitioned into several wavelengths (w) using Wavelength Division Multiplexing (WDM).

In this paper, we model one such output fiber with its own set of wavelength converters. Each outgoing wavelength λ_i, $i = 1, 2, \ldots, w$, has k number of incoming wavelengths λ_i, one per input fiber, targeted into it. Part of the bursts arriving in each of the k input wavelengths λ_i is switched to the destination wavelength λ_i of the output fiber under study. The remaining bursts are switched through other output fibers. An incoming burst on λ_i will try to be scheduled on its *home* wavelength λ_i in the outgoing fiber. In case the home wavelength is busy, the burst tries to occupy adjacent wavelengths in the range d, i.e $\lambda_i \pm d$, if a converter in the common converter pool per output fiber is available. The method of trying to occupy an alternate wavelength will be explained in the analytical model section.

3 The Queueing Model for Small *d*

The Markov process for limited-range wavelength conversion does not have a product-form solution [9]. In view of this, we propose an approximate solution for the problem of computing blocking probabilities in an OBS switch that employs limited-range wavelength conversion. The queueing problem is modeled as a simultaneous resource possession problem as follows:

Let us consider a three wavelength example to illustrate our method. The original model is decomposed into sub-systems as depicted schematically in figure 1.

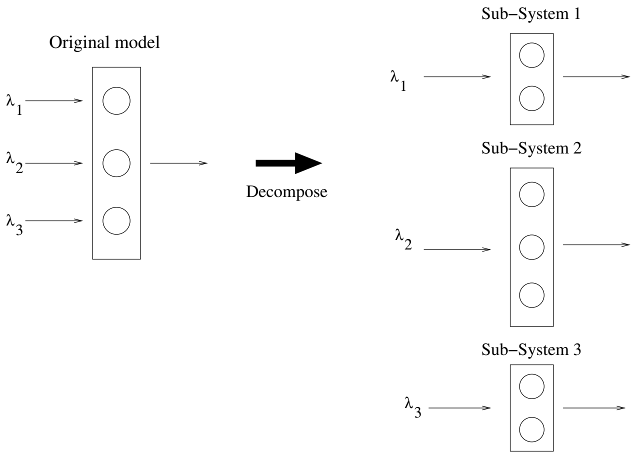


Fig. 1. A three node model

Figure 1 depicts the decomposition of a system with $w = 3$ and $d = 1$ into three sub-systems $\overline{n}_1$,$\overline{n}_2$ and $\overline{n}_3$ each being an Erlang loss queue. The number of sub-systems is equal to the number of wavelengths w. Each sub-system $\overline{n}_i$ represents a *home* wavelength λ_i and neighboring wavelengths into which a burst coming on the home wavelength can be converted into. We represent each wavelength with a server in the Erlang loss queue sub-system. Thus, each sub-system has number of servers ranging from $(d+1)$ to $(2*d+1)$. The border sub-systems $\overline{n}_1$ and $\overline{n}_3$ have two servers and the central sub-system $\overline{n}_2$ has three servers. The state of each server is represented by $n_{i,j}\epsilon\{0,1\}$, where i is the sub-system and j is the wavelength index. Thus, $\overline{n}_1 = \{n_{1,1}, n_{1,2}\}$, $\overline{n}_2 = \{n_{2,1}, n_{2,2}, n_{2,3}\}$ and $\overline{n}_3 = \{n_{3,2}, n_{3,3}\}$.

The state of the system is given by the union of states of servers in all the sub-systems combined. Thus, for the three node example, the state of the system is represented by the tuple $(\overline{n}_1, \overline{n}_2, \overline{n}_3) = (n_{1,1}, n_{1,2}, n_{2,1}, n_{2,2}, n_{2,3}, n_{3,2}, n_{3,3})$.

Let $p(\overline{n}_1, \overline{n}_2, \overline{n}_3)$ be the probability of the system being in one such state. Then, we have

$$p(\overline{n}_1, \overline{n}_2, \overline{n}_3) = \frac{1}{G} * p(\overline{n}_1) * p(\overline{n}_2) * p(\overline{n}_3) \tag{1}$$

or,

$$\begin{aligned} p(n_{1,1}, n_{1,2}, n_{2,1}, n_{2,2}, n_{2,3}, n_{3,2}, n_{3,3}) = \frac{1}{G} * p(n_{1,1}, n_{1,2}) \ & *p(n_{2,1}, n_{2,2}, n_{2,3}) \\ & *p(n_{3,2}, n_{3,3}) \end{aligned} \tag{2}$$

where $G = \sum p(\overline{n}_1) * p(\overline{n}_2) * p(\overline{n}_3)$ summed over all feasible states.

The problem has been decomposed to a granularity which allows us to apply the rules of simultaneous resource possession.

Rule 1: A wavelength server can be occupied in only one sub-system
Rule 2: The number of conversions cannot exceed c

Thus we have,

$$n_{1,1} + n_{2,1} = 1 \tag{3}$$

$$n_{1,2} + n_{2,2} + n_{3,2} = 1 \tag{4}$$

$$n_{2,3} + n_{3,3} = 1 \tag{5}$$

$$n_{1,2} + n_{2,1} + n_{2,3} + n_{3,2} \leq c \tag{6}$$

The wavelength occupancy constraint (*Rule 1*) and the converters constraint (*Rule 2*) reduce the state space from which the solution is computed. The computation of the individual sub-system probabilities $p(\overline{n}_x)$, where $x \epsilon \{1 \ldots w\}$ will be explained in the sub-section 3.2.

When we generalize our model to any number of wavelengths w, we have,

$$p(\overline{n}_1, \overline{n}_2, \overline{n}_3, \ldots, \overline{n}_w) = \frac{1}{G} * p(\overline{n}_1) * p(\overline{n}_2) * p(\overline{n}_3) * \ldots * p(\overline{n}_w) \tag{7}$$

The model is further decomposed so as to include the constraints of wavelength occupancy and the number of available converters. Each subsystem thus becomes:

$$\overline{n}_i = (n_{i,i-d}, n_{i,i-d+1}, ..., n_{i,i}, ..., n_{i,i+d-1}, n_{i,i+d}) \tag{8}$$

and in $n_{i,j}$, i,j $\epsilon \{1 \ldots w\}$

The constraints for simultaneous resource possession are:

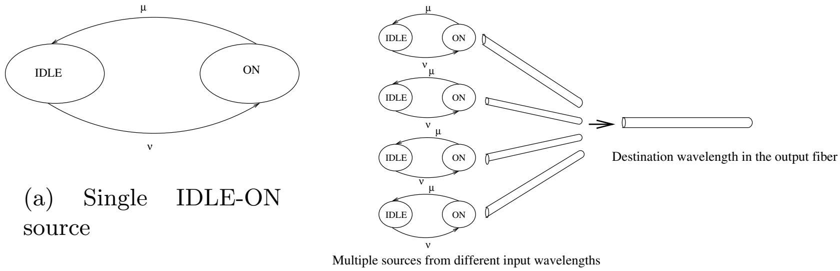
$$n_{i,i-d} + n_{i,i-d+1} + \ldots + n_{i,i-1} + n_{i,i} + n_{i,i+1} + \ldots + n_{i,i+d-1} + n_{i,i+d} = 1 \forall i = 1, 2, .., w \tag{9}$$

$$\sum_{i=1,j=i-d,j\geq 1}^{i=w,j=i+d} n_{i,j} \leq c \tag{10}$$

The computation of G is a complicated task because of the state space explosion with large values of w and d. We shall describe the approach we take for the computation of G in sub-section 3.3. Once the probability of the system existing in each state has been determined, the blocking probability of each wavelength is then the sum of all the corresponding blocking states.

3.1 The Arrival Process

We use the IDLE-ON arrival process shown in figure 2 to generate bursts on a single incoming wavelength λ_i. The IDLE and ON periods are exponentially distributed with a mean of $\frac{1}{\nu}$ and $\frac{1}{\mu}$ respectively. A single ON period generates a single burst. An IDLE period is followed by the ON period and vice-versa.


μ
IDLE
ON
ν
Destination wavelength in the output fiber
Multiple sources from different input wavelengths

(a) Single IDLE-ON source

(b) Multiple IDLE-ON sources

Fig. 2. Arrival process

If a burst is dropped, the source returns to the IDLE state. Since there are k input fibers, the burst arrival process from all the k incoming wavelengths λ_i is the superposition of k single IDLE-ON sources as shown in figure 2(b). We assume that all the k arrival IDLE-ON sources are identical. We also assume in this paper that the superposition process is identical for all wavelengths λ_i, $i = 1 \ldots w$.

As described in section 2, we are modeling a single output fiber. Each wavelength in the fiber will only have some of the traffic directed towards it from its k corresponding input wavelengths. Consequently, the IDLE period is assumed to have been appropriately extended so that only the bursts destined to the outgoing wavelength in the fiber are modeled.

3.2 Computing the Probabilities for Each Sub-system

In each sub-system, a customer is allocated to the home server. In case it is busy, a free immediate adjacent server is chosen with probability 0.5. If both are busy, the next set of adjacent servers are scanned for an empty server. The probabilities for each sub-system can be obtained from the birth-death process and lacks a closed-from expression [9]. On increasing the values of w and d, it gets difficult to solve a Markov chain for each sub-system. On close examination, it can be seen that adding a few transitions to the birth-death can get us a closed-form expression, which is the Erlang system or the Engset system depending on the assumption regarding the arrival process. By using either of these systems, we can get away with the problem of marking the servers. Thus, we can assume that the servers are allocated randomly. This gives rise to a product-form solution which is an approximation to the probability terms $p(\overline{n}_x)$ in equation 7.

In equation 2, the probability terms on the right hand side are determined using the Engset model. If n is the number of servers, k the number of input fibers and μ, ν are as described in the arrival process, the probability that there are i customers in such a $M/M/n/n/k$ system is given by:

$$\pi_i^* = \frac{\binom{k-1}{i}(\frac{\nu}{\mu})^i}{\sum_{j=0}^{n}\binom{k-1}{j}(\frac{\nu}{\mu})^j} \tag{11}$$

The Engset probabilities of equation 11 are those as seen by an arrival.

The approximated probabilities that a specific set of i servers is occupied in a sub-system is given by:

$$\pi_i = \frac{\pi_i^*}{\binom{n}{i}} \tag{12}$$

The product of such terms if the state is valid can be added to compute G. The blocking probability of a particular wavelength is finally computed by summing up all the appropriate blocking states.

3.3 Computation of G

We now describe the approach taken to compute the normalization constant G. By brute-force enumeration, it takes O(w^w) to cover the entire state space and determine the blocking probabilities for each wavelength. It is clear that brute-force enumeration cannot be used to compute G for large values of w and d. We have determined that this approach can be taken to solve for $d = 1, 2$ and very small values of w. For larger values of d, we propose the *large scale approximation technique* described in section 4.

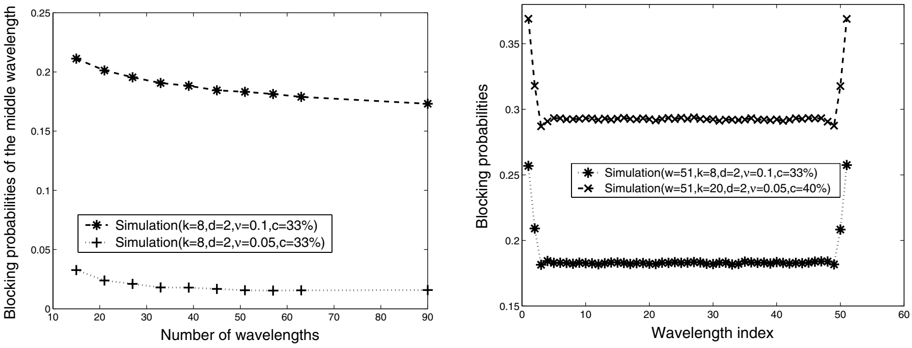
3.4 Large Number of Wavelengths

In this section, we describe the method for extending the computation of blocking probabilities for large number of wavelengths where $d = 1, 2$.

As was described in section 3.3, we can compute G by brute-force enumeration only for very low values of d and w. However, there are some properties of blocking probabilities with increasing number of wavelengths that we can make use of to compute blocking probabilities for these large systems.

Figure 3(a) plots the simulation results for the blocking probability of the middle wavelength of the system with increasing number of w in the system. The number of converters increases proportionally with the number of wavelengths. It can be seen that with increase of w, there is a very small gradual decrease in the blocking probability. It can also be seen that the trend is similar for different arrival rates. We recall that we have assumed an identical superimposed arrival process to each wavelength.

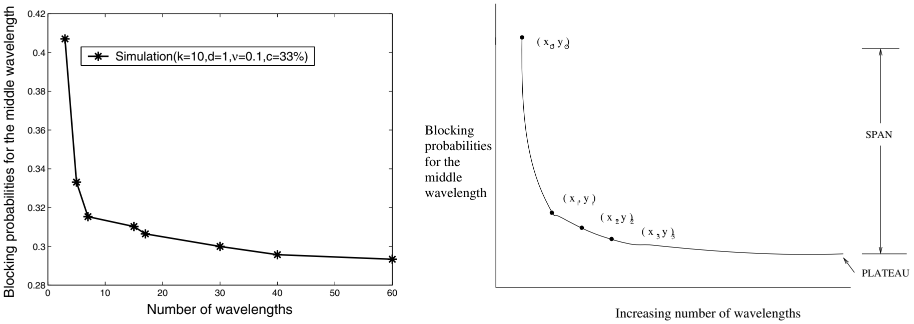
Figure 3(b) plots the simulation results for the blocking probabilities for all the wavelengths in the system, for two different cases. We assume that each wavelength is fed by the same superposition arrival process. It can be seen that the border wavelengths have the highest blocking since the degree of conversion is smaller for them compared to the center wavelengths. In the following sections, we focus only the blocking probabilities of the middle wavelength and it can be noted that the border wavelengths have higher values. Also, we assume symmetric traffic wherein each wavelength is fed by the same arrival process resulting in the same arrival rate.



(a) proportional use of converters

(b) Blocking probabilities for the complete spectrum

Fig. 3. Large number of wavelengths



(a) Exponential decay

(b) Four-point method

Fig. 4. Large number of wavelengths

3.5 The Four-Point Method

From the previous section, it can be seen that a smaller system (smaller w) with a proportionally smaller number of converters can be made use of to get an approximate value of the blocking probabilities, which then slowly decreases with increasing w.

Figure 4(a) plots the simulation results for the blocking probability of the middle wavelength with increasing number of wavelengths as in figure 3(a), but starting with very few wavelengths. It can be seen that the distribution is very similar to the exponential distribution. In the four-point method, we use the brute-force enumeration described in section 3.3 to calculate the initial 4 points on the curve. We then use these points to determine the value of the plateau P of the curve, see figure 4(b).

Let $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ and (x_3, y_3) be the initial 4 points on the curve. The size of the system W is $(2 * d + 1)$, $(2 * d + 3)$, $(2 * d + 5)$ and $(2 * d + 7)$ respectively. The converters are proportionally decreased from their initial value in the original system w. A one phase exponential decay function with the initial 4 points can be used to determine P, see [6], which is the blocking probability for large number of w and for $d = 1, 2$.

4 Large Scale Approximation

The methods discussed in section 3 can only be applied for $d = 1, 2$. In this section, we describe an approximate model for large d. The schematics of the model are shown in figure 5.

The system is decomposed into two loss queues. The first one, referred to as the *server bottleneck* and the second one as the *converter bottleneck*. The server (i.e. wavelengths) bottleneck is used to determine the blocking due to lack of servers for a given d, and the converter bottleneck is used to determine blocking due to lack of converters c. The server bottleneck and the converter bottleneck are modeled separately and independence is assumed between them.

Figure 6 forms the basis for modeling the server bottleneck. The original system of size w and d with limited-range conversion has been transformed to a modified system of size $(2 * d + 1)$ with full-range conversion. As was stated earlier in subsection 3.5, the effect of border wavelengths tends to decrease as the system size considered, i.e the number of wavelengths w, increases. It was observed that this increase of system size can be approximately modeled by conversion from limited-range to full-range with $w = (2 * d + 1)$. Further, the number of converters for the $(2 * d + 1)$ system is proportionally reduced from the original system. A numerical example of this is shown in figure 7, where

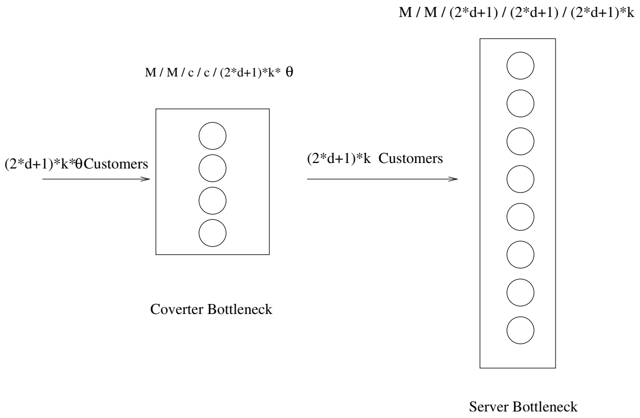


Fig. 5. Large scale approximation

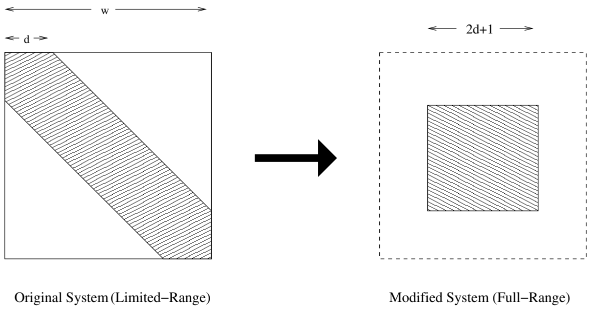


Fig. 6. Conversion from limited-range to full-range conversion

the y axis denotes the *average blocking probability* of all the wavelengths. We use the same superposed arrival process to each wavelength as was described in section 3.1.

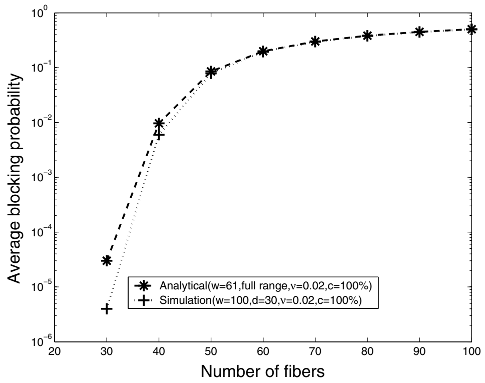


Fig. 7. Full-range conversion

The server bottleneck is modeled by an Engset system $M/M/(2*d+1)/(2*d+1)/((2*d+1)*k)$. The converter bottleneck is also modeled as an Engset system. But the percentage of $((2*d+1)*k)$ customers, θ, that have to use the converter pool needs to be determined. The approximate θ is determined as follows:

$$\begin{aligned}&\text{probability (an arrival using a converter)}\\&= \text{probability (its destination wavelength is busy)} = p_{Blocking}(M/M/1/1/k)\end{aligned}$$

Therefore,

$$\begin{aligned}\text{No. of arrivals that use a converter} &= (2*d+1)*k*p_{Blocking}(M/M/1/1/k)\\&= (2*d+1)*k*\theta\end{aligned}$$

Thus, the Engset system for converter bottleneck becomes

$$M/M/c/c/((2*d+1)*k*\theta).$$

If p_{Bs} is the probability that a burst gets blocked at the servers and p_{Bc} is the probability that a burst gets blocked at the converter pool, then, because of the independence assumption the probability that a burst gets blocked is given by:

$$p_B = 1 - (1 - p_{Bs})(1 - p_{Bc}) \tag{13}$$

5 Results

An event based simulation model was constructed to evaluate the results of the analytical model. The simulation results were plotted with 95% confidence interval estimated by the method of batch means, see Perros [7]. Each batch is completed when every wavelength has 30,000 bursts arriving at it. The confidence intervals are very tight and are not discernible in the graphs. A software called *GraphPad Prism*[1] was used to fit a one phase exponential decay function with the initial 4 points as described in the four-point method.

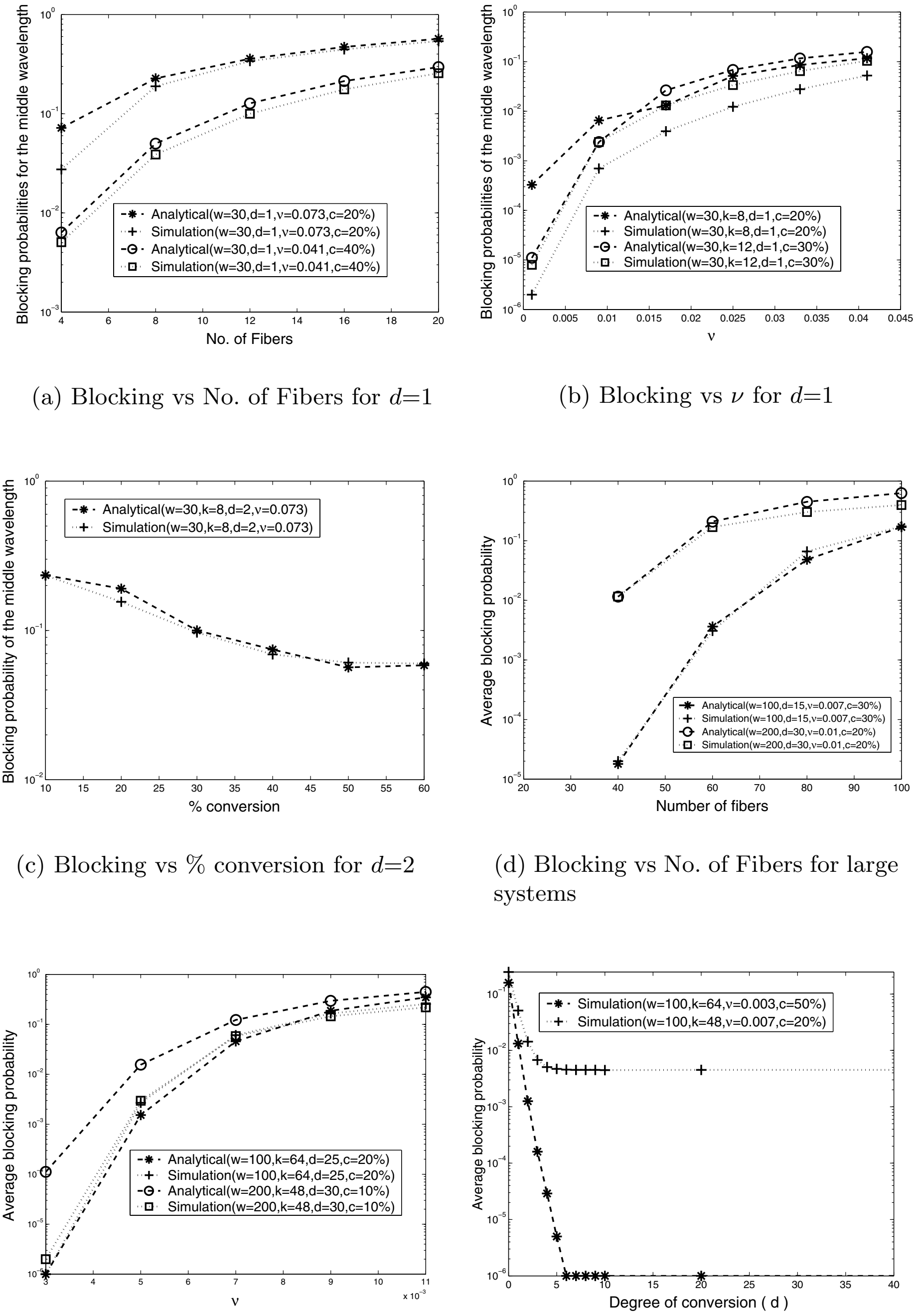
The blocking probability of the center wavelength was calculated by varying ν (which varies the arrival rate), the number of fibers k, the degree of conversion, d and the number of wavelengths, w. Both analytical results and simulation results have been plotted. We use the four-point method for $d = 1, 2$ and large scale approximation for larger d. Only a subset of these results are shown here. Please refer to [9] for the complete set.

The way we take the initial 4 points differs for $d = 2$. To get the fourth point, we have to consider a system of size $(2*d+7) = 11$. By brute-force enumeration, it is not possible to explore all the states of such a system within reasonable time. So, we get the fourth point via simulation, which takes far lesser time. We note that the analytical values are much closer to the simulation ones than in the case of $d = 1$. However, the range for very good accuracy seem to be between 10^{-1} and 1. For lower blocking probabilities it is hard to fit a curve because of the gap between the three analytical points and the fourth simulation point.

Figure 8(f) plots the simulation results for large values of d. It can be seen that increasing d is most effective at low arrival rates and/or at high % conversion. Increasing d does not have a big impact when blocking probabilities are high.

6 Conclusion

In this paper, we proposed an approximate solution for limited-range wavelength conversion in an OBS switch. The problem was modeled as a simultaneous resource possession problem and an approximate product-form solution was proposed. This solution could be applied for very small values of w and d. A method called the four-point method extended the solution for larger values of w. We then proposed a large scale approximation technique which calculated average



(a) Blocking vs No. of Fibers for d=1

(b) Blocking vs ν for d=1

(c) Blocking vs % conversion for d=2

(d) Blocking vs No. of Fibers for large systems

(e) Blocking vs ν for large systems

(f) Blocking vs Degree of conversion (d) for large systems blocking probabilities for very large values of w and d. d has significant impact on blocking when the percentage of converters available is high and/or low arrival rates. When the percentage of conversion is small, the benefits of d saturates very early and large values of d may not be useful.

References

1. *GraphPad Prism.* Available for downloading at http://www.graphpad.com, 2004.
2. N. Akar and E. Karasan. Exact calculation of blocking probabilities for bufferless optical burst switched links with partial wavelength conversion. *In Proceedings of Broadnets*, October 2004.
3. L. Battestilli and H. Perros. An introduction to optical burst switching. *IEEE communications magazine*, 41:S10–S15, 2003.
4. J. Yates, J. Lacey, D. Everitt and M. Summerfield. Limited-range wavelength translation in all-optical networks. *In Proceedings of IEEE INFOCOM*, 3:954–961, March 1996.
5. C. Gauger. Optimized combination of converter pools and FDL buffers for contention resolution in optical burst switching. *Photonic network communications*, 8:139–148, 2004.
6. H. Motulsky and A. Christopoulos. *Fitting Models to Biological Data using Linear and Nonlinear Regression: A practical guide to curve fitting.* Available for free download from http://www.curvefit.com/, 2003.
7. H. G. Perros. *Computer Simulation Techniques: The definitive introduction.* Available for free download from http://www.csc.ncsu.edu/faculty/perros/hp.html, 2004.
8. I. Baldine, G. Rouskas, H. Perros and D. Stevenson. Jumpstart: a just-in-time signaling architecture for WDM burst-switched networks. *IEEE communications magazine*, 40:82–89, 2002.
9. V. Puttasubbappa and H. Perros. *An Approximate Queueing Model for Limited-Range Conversion in an OBS Switch.* Available at http://www.csc.ncsu.edu/ faculty/perros/Vishwas3.pdf, 2004.
10. X. Qin and Y. Yang. Nonblocking WDM switching networks with full and limited wavelength conversion. *IEEE Transactions on Communications*, 50:2032–2041, December 2002.
11. J. Ramamirtham and J.Turner. Design of wavelength converting switches for optical burst switching. *In Proceedings of IEEE INFOCOM*, 1:2032–2041, June 2002.
12. T. Tripathi and K.N. Sivarajan. Computing approximate blocking probabilities in wavelength routed all-optical networks with limited-range wavelength conversion. *In Proceedings of IEEE INFOCOM*, 1:329–336, March 1999.
13. Y. Xiong, M. Vandenhoute, and H. Cankaya. Control architecture in optical burst-switched WDM networks. *IEEE Journal on selected areas in communications*, 18:1838–1851, October 2000.
14. Z. Rosberg, H. Vu and M. Zukerman. Performance evaluation of optical burst switching networks with limited wavelength conversion. *In Proceedings of ONDM 2003, The 7th IFIP Working Conference on Optical Network Design and Modeling*, 2:1155–1169, February 2003.
15. J.Y. Wei and R.I. McFarland. Just-in-time signaling for WDM optical burst switching networks. *Journal of Lightwave Technology*, 18:2019–2037, December 2000.

On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks

Saurabh Tewari and Leonard Kleinrock

Computer Science Department, University of California at Los Angeles,
4403 Boelter Hall, Los Angeles, CA 90095, U.S.A.
{stewari, lk}@cs.ucla.edu

Abstract. Each peer in a peer-to-peer network, by definition, is both a consumer and a provider of the service. As a consumer, a peer wants to obtain its objects of interest as quickly as possible. However, as a service provider, the peer wants to serve no more than an equitable portion of the total workload. Our first observation in this paper is that if one satisfies the latter criterion of fairness in workload distribution, then one also minimizes the average download time when the delay at the server is convex in the utilization factor of the server. We had previously observed that controlled flooding search in unstructured networks is optimized when the number of replicas of a file is proportional to the request rate for that file. Here we show that such a replica distribution also ensures fairness in the workload distribution and, at the same time, minimizes the average download time seen by a download request.

1 Introduction

Peer-to-peer networks offer a promise of systems that automatically scale in capacity as the number of users increases and yet are extremely robust, automatically adapting to failures of nodes/links as well as to changes in usage patterns, all at virtually no cost. These loosely organized networks of autonomous entities (user nodes or "peers"), which make their resources available to other peers, represent a new computing paradigm where the service consumers are, now, the service providers as well. These systems offer the potential of a tremendous improvement over the traditional client-server architectures.

For a user of a peer-to-peer content distribution system, the measure of the system performance is the time it takes to fulfill request for a particular file which consists of two components: the time it takes to find who has that content, and the time to actually download the content. While a significant amount of attention has been paid to the search aspect in past, as the sizes of the files exchanged become larger and larger, the download time has become the dominant factor in the perceived performance of the system by a user. This shift towards larger file sizes also implies that resources consumed in supportm of file searches is no longer the dominant component of the peer-to-peer workload at a peer but, rather, servicing the download requests becomes the majority of the workload. Thus, as a service consumer, a peer wants to see the minimum possible download time, and as a service provider, the peer

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 709 – 717, 2005.

expects a fair distribution of the workload. In this paper, we address these objectives of fairness in workload distribution and minimization of the download time. After briefly discussing the related work, we present our model for the peer-to-peer system in Section 3. Optimization of the download time is discussed in Section 4 where we establish that our two objectives are perfectly congruent as uniform node utilization also minimizes the download time. Section 5 discusses uniform node utilization where we show that if the number of replicas of a file is proportional to the request rate for that file, each node operates at an equal utilization factor regardless of which files it is sharing, and hence, such a proportional file replication also minimizes the download time. Section 6 contains our conclusions.

2 Related Work

As the size of content exchanged over peer-to-peer networks has increased from few megabytes to hundreds of megabytes, a few researchers have begun to look the issue of download performance. [10] and [14] model the download service capacity in the BitTorrent file-sharing system [2] in relation to the evolution of the file request rate (specifically, they derive the download service capacity of the system as a function of the age of the file in the system) while [12] studies the download time in these systems via simulations. However, since all of them seek to model the performance of a particular system, they do not attempt to optimize the download performance. [4] presents a fairly detailed model for peer-to-peer file sharing systems that incorporates the effect of both the search-processing load as well as the download service loads at nodes and the download service capacities for each file for peer-to-peer file systems using different search mechanisms. However, like [10] and [14], its objective is also modeling of existing designs and they do not examine alternatives that may improve performance. Our peer-to-peer system model is much simpler which gives us the advantage that we can find opportunities for performance optimization with a relatively simple analysis.

In terms of similarity in peer-to-peer system models and interest in file replication distribution as a mean to improve system performance, [9] and [13] are closely related to this paper. However, these papers address the search performance in unstructured peer-to-peer networks so their models include additional search-related constructs whereas our results are independent of any underlying search mechanism. We use the file replication distribution shown to be optimal for controlled flooding search in [13] as our starting point as we seek a replication distribution to minimize the download time. While [9] finds the proportional file replication to be sub-optimal for their preferred random walk search mechanism, in rejecting this solution it did note that such a distribution implies that each replica of a file services equal download requests per unit time. However, due to their focus on search performance, [9] did not take the next step of finding that this distribution implies uniform node utilization. Further, since it does not address the download time, the optimality of such a distribution for download time is overlooked. Further, our results are applicable to structured peer-to-peer systems as well. File replication is addressed in the context of structured peer-to-peer systems by [5] and [8] among others; while [5] addresses lookup performance, [8] is similar to our work in that it addresses the download performance. However it does not seek to explicitly optimize the download time.

Many researchers overlook file replication distribution as an option for improving system performance since the number of replicas of a file in currently deployed systems is not a control parameter but a byproduct of user requests and the downloading process [15]. However, as [9] and [13] discuss, near-optimal replication distributions can be automatically achieved using distributed algorithms that preserve the underlying user request and the download process characteristics.

3 Model

Our peer-to-peer system model consists of *nodes* and *files*. The term *files* represents any generic content while a node corresponds to a user or peer (these terms are used interchangeably in this paper). Each file has a certain request rate associated with it, reflecting user interest in that file. A file can have more than one replica in the system. Nodes have finite local storage space to store file replicas. If a user "requests" a file and if the file is not present in its local storage, the file is downloaded from one or more copies of that file in the network. While our model is independent of the specifics of the search mechanism used by the nodes to determine which peers they can download the file from, we do assume that the download requests for a file are uniformly distributed over all the replicas of that file[1]. Finally, we also assume that a node will always satisfy a request for a file present in its local storage[2]. We make the following two additional assumptions for ease of discussion:

Assumption 1. Each node is homogenous in service capacity.
Assumption 2. Each file is of equal size.

We note here that measurement studies of existing peer-to-peer file sharing systems show considerable variation in file sizes and the service capacity (link bandwidth) of peers [11]. However, our focus is on design alternatives in general peer-to-peer systems and alternative peer-to-peer applications may not share these characteristics of file-sharing systems. In a "true" peer-to-peer system, the peers contribute equal resources to the system so it is reasonable to assume that the service capacities of peers are equal. One can incorporate the cases of unequal file sizes and unequal server capacities in the analysis in this paper to obtain analogous results for the current peer-to-peer file sharing systems if desired. In fact, our analysis and the

[1] This assumption should hold if the search mechanism directs the nodes requesting the file to the nodes that have the file in a uniform manner and/or if the requesting nodes download the file from all nodes that have the file in parallel.

[2] This does not imply that all nodes share files but only that if a node makes a file available, it will satisfy a download request for the file (whereas if a node has a particular file but does not make it available, it will never get a search request for it). In this sense, the number of replicas of a file in the system should be considered as the number of *available* replicas of the file in the system.

derived results directly applies to the special case of non-homogenous peer service capacities observed in current file sharing systems where a significant fraction of peers do not contribute any resources, a phenomenon generally referred to as *free-riding* [11]. Presence of such users just decreases the overall download service capacity available but the fairness of workload distribution among the sharing nodes and optimality of the download performance is maintained; only the load on the sharing nodes increases[3]. We use the following notation for the system parameters:

M = number of nodes in the system
N = number of unique files in the system
K = per-node storage size in number of files
μ =download service capacity of a node (files/unit time)
λ_i = request rate for file i per node[4] (requests/unit time)
$\lambda = \sum_{i=1}^{N} \lambda_i$
n_i = number of replicas of file i in the system

As discussed earlier, the metrics of interest are the variation in the service load at each node and the average download time for a file request.

The service load Λ_j at node j that has files $R_j = \{r_1, r_2, r_3, \ldots, r_K\}$ where $r_i \in \{1, 2, 3, \ldots, N\}$ $\forall i$, in its storage is given by:

$$\Lambda_j = \sum_{i \in R_j} \frac{M\lambda_i}{n_i} . \tag{1}$$

since, given our assumption that the $M\lambda_i$ requests (per unit time) for file i are uniformly distributed over the n_i replicas of file i, each replica of file i serves $M\lambda_i/n_i$ requests per unit time.

The download time for a file depends on three main factors: the size of the file, the traffic on the underlying network layer, and how fast the node(s) serving the content (henceforth, referred to as the server) can upload the file. In practice, however, the upload link bandwidth of the node serving the content is the predominant constraint in downloads. Besides, including the effect of the traffic on the underlying network layer is difficult as downloads are not the only thing carried on the underlying network layer. Therefore, we approximate the download time for equal-sized files by the delay experienced by a download request at the server. For a server to upload the desired content, the upload link as well as the server processing cycles to feed the link should be available. Thus, the delay experienced at the server depends on the overall workload on the server of which the download request rate serviced is only a part. We

[3] This can be seen with the modifications to our analysis of Sections 4 and 5 as follows. If a fraction, $1-\gamma$, of the nodes do not service any downloads, the total storage capacity of the system goes down to γKM and, with proportional file replication, each of the sharing nodes now services λ/γ download requests per unit time, on average. Since each of the sharing nodes operates at uniform utilization level, the average download time is optimal.

[4] λ_i is the request rate for file i in the system averaged over all M nodes i.e. $\lambda_i = \frac{\sum_{j=1}^{M} \lambda_{ij}}{M}$.

do not explicitly model the effect of interruptions by other processes running on the server but rather include them in the randomness of the delay distribution function. Finally, whenever we talk of download time in this paper, it is with the understanding that for larger size files, we seek the download time performance for fixed-size file blocks. Hence, in our model the delay seen by a request at a content server only depends on the download request rate at the server and the download service capacity of the server. To simplify the analysis, we further assume that the delay distribution function at each server is identical. Thus, the average download time for a request serviced by a node j running at utilization factor Λ_j /μ is $T(\Lambda_j /\mu)$. Therefore, the average download time τ for a file request is given by:

$$\tau = \sum_{j=1}^{M} \frac{\Lambda_j}{M\lambda} T(\frac{\Lambda_j}{\mu}) . \tag{2}$$

where Λ_j is as described by Eq. (1), $M\lambda$ is the total download request rate in the system and $T(\rho)$ is the delay distribution at a server running at utilization factor ρ.

4 Download Time

The desirability of load-balancing multiple homogenous servers providing a service is well known[5] (although some exceptions have been noted in literature [3]). Formally, the optimality of operating each node in the system at the same utilization factor requires that the service time at the server be convex in the utilization factor at the server (which is true for majority of the queueing systems [7]). In the context of our system, we can see this as follows.

We can rewrite Eq. (2) as follows:

$$\tau = \frac{\mu}{M\lambda} \sum_{j=1}^{M} \rho_j T(\rho_j) .$$

where $\rho_j = \Lambda_j /\mu$. Since $T(\rho)$ is convex, $\Phi(\rho) = \rho T(\rho)$ is also convex. Using convexity of $\Phi(\rho)$[6] [6]:

$$\Phi\left(\frac{\rho_1 + \rho_2 + \rho_3 + \cdots + \rho_M}{M}\right) \le \frac{\Phi(\rho_1) + \Phi(\rho_2) + \Phi(\rho_3) + \cdots + \Phi(\rho_M)}{M}$$

For ease of the remaining discussion, we state this result as applied to our peer-to-peer file sharing system as the following theorem.

Theorem 1. If the download time $T(\rho)$ at a node is convex in the utilization factor ρ at the node, the average download time in the system is minimized with uniform node

[5] For recent evidence, one only need look at the large number of commercial as well as open-source load-balancers available for data center applications where multiple web servers are used to support popular web sites.

[6] Also referred to as the extension of *Jensen's Inequality* to convex combinations of more than two points [1].

utilization (all nodes in the system operating at an equal utilization factor) when the delay distribution function *T(ρ)* is identical for all nodes and each node has identical service capacity.

While the desire for uniform node utilization in multiple server systems is driven solely by performance optimization, in peer-to-peer systems it has significance much beyond the download performance optimization. As discussed earlier, in peer-to-peer systems, each peer is an independent entity and fairness in the upload contribution asked of each peer is an important objective.

5 Uniform Node Utilization

The optimal replica distribution for controlled flooding search in unstructured peer-to-peer systems has been found [13] to be one where the number of replicas of each file is proportional to the request rate for that file, i.e. $n_i \propto \lambda_i \; \forall i$ where n_i and λ_i are as defined in Section 3. In this section, we find that the above proportional replication also achieves uniform node utilization and, hence, optimal download time.

Theorem 2. If the download requests for a file are uniformly distributed over all the replicas of the file in the system, each node operates at the same utilization factor, independent of the files it has in its storage, when the number of replicas of file *i*, n_i, is proportional to the average request rate for file *i*, λ_i, for all files, i.e.

$$n_i \propto \lambda_i \, i \,. \tag{3}$$

when each file is of equal size.

Proof:
Given *M* nodes in the system, each with the capacity to store *K* files, the total amount of storage available in the system is *MK*. Since $\lambda = \sum_{i=1}^{N} \lambda_i$ is the total request rate in the system averaged over the *M* nodes, $n_i \propto \lambda_i \; \forall i$ implies that:

$$n_i = \frac{\lambda_i}{\lambda} KM \quad i \,. \tag{4}$$

As noted in Section 3, each replica of file *i* serves $M\lambda_i/n_i$ requests per unit time if the download requests for files are uniformly distributed over all the replicas of the files. Therefore, for the replica distribution defined by Eq. (4), each replica of file *i* serves λ/K requests per unit time. Since each node would clearly keep as many files as possible locally, as long as $K < N$, i.e. the per-storage size *K* is not sufficient to store all the possible *N* files in the system, each node has *K* files in its storage. Therefore, the download request rate served by each node is:

$$\Lambda_j = \sum_{i \in R_j} \frac{M\lambda_i}{n_i} = \sum_{i \in R_j} \frac{\lambda}{K} = \lambda \,.$$

Thus, each node serves λ download requests per unit time, the same as the download request rate injected by a node on average. Hence, when the number of replicas of each file is proportional to the request rate for that file, each node, on average, serves an equal number of download requests per unit time. Since each node is assumed to have an equal download service capacity, each node has the same node utilization factor.

Q.E.D.

From Theorems 1 and 2, it directly follows that:

Corollary 1. If the delay distribution function *T(ρ)* at a node operating at utilization factor ρ is convex and identical at each node, and the download requests for a file are uniformly distributed over all the replicas of the file in the system, the average download time in the system is minimized when the number of replicas of file *i*, n_i, is proportional to the average request rate for file *i*, λ_i, for all files, i.e.

$$n_i \propto \lambda_I \quad i\,.$$

under Assumptions 1 and 2.

Notice that while the uniform node utilization condition in Theorem 1 is both necessary and sufficient for download time optimization, the proportional file replication of Theorem 2 is only a sufficient condition as we did not show that such a distribution is the only solution for uniform node utilization. However, such a solution is attractive as it is independent of which files a node keeps in its local storage. Thus, even if the files present at a node keep changing (depending, for example, on the requests made more recently or more frequently) but as long as the number of replicas in the overall system stays proportional to the request rate for the file, the average download time remains optimal. Thus, for example, in a peer-to-peer system where a consequence of finite per-node storage is the possibility of deletion of old files to make space for newly requested files, if the per-node storage management algorithm leads to proportional replication at equilibrium, one has an autonomic content delivery system that automatically adjusts the resource allocation to the optimal setting even as the file request patterns evolve. [13], in its presentation of optimal file replication for controlled flooding search in unstructured peer-to-peer networks, includes such an example system where LRU storage management leads to near-proportional file replication. [9] also discusses distributed algorithms in a similar peer-to-peer system model although they sought to achieve a square-root replication (i.e. $n_i \propto \sqrt{\lambda_i}$) distribution which is optimal for random walk search in unstructured peer-to-peer networks. We emphasize here that our result that the proportional file replication minimizes the download time while, at the same time, ensuring fairness in workload distribution is independent of any search mechanism. Thus our result is applicable to both structured as well as unstructured peer-to-peer systems as long as the underlying system directs the download requests for a file uniformly over all the replicas of that file. Further, this result is independent of any specific file popularity distribution and does not assume any specific request arrival process or service time distribution. Our only assumption is that the download request arrival process and the download service time distribution is such that the delay distribution at a server is convex in the server utilization factor; this is not unduly restrictive as a majority of queueing systems have convex delay distribution [7].

6 Conclusions

In this paper, we address the issue of fairness in the workload distribution and optimization of average download time in peer-to-peer systems. Our first observation is that the download time is minimized when each peer is operating (in terms of download requests it services) at an equal utilization factor. Next, we show that when the number of replicas of each file is proportional to the request rate for that file, each peer operates at a uniform utilization level independent of which files it has in its storage. This result on proportional file replication implying equal workload distribution and optimal download time is independent of any search mechanism and we only assume that the download requests for a file are uniformly distributed over all the replicas of that file in the system. The optimality of the proportional file replication for the average download time also does not depend on any specific request arrival process or service time distribution, but only on the assumption that the delay distribution for a download request is convex in the utilization factor of the peer servicing that request.

References

1. Boyd, S., and Vandenberghe, L.: Convex Optimization, Cambridge University Press, London, 2004.
2. Cohen, B.: Incentives Build Robustness in BitTorrent. In Proceedings of the Workshop on Economics of Peer-to-Peer Systems, 2003.
3. Crovella, M., Harchol-Balter, M., Murta, C.: Task Assignment in a Distributed System: Improving Performance by Unbalancing Load. In Proceedings of ACM SIGMETRICS 1998.
4. Ge, Z., Figueiredo, D. R., Jaiswal, S., Kurose, J., Towsley, D.: Modeling peer-peer file sharing systems. In Proceedings of IEEE Infocom 2003.
5. Gopalakrishnan, V., Silaghi, B., Bhattacharjee, B., Keleher, P.: Adaptive replication in peer-to-peer systems. In Proceedings of the 24th ICSDCS, 2004.
6. Hardy, G. H., Littlewood, J. E., Pólya, G. Inequalities, Cambridge University Press, London, 1952.
7. Kleinrock, L.: Queueing Systems, Volume I: Theory. Wiley Interscience, New York, 1975.
8. Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: An Architecture for Global-scale Persistent Storage. In Proceedings of the 9th ASPLOS, 2000.
9. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S. Search and Replication in Unstructured Peer-to-Peer Networks. In Proceedings of the 16th ACM ICS 2002.
10. Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In Proceedings of ACM SIGCOMM 2004. .
11. Saroiu, S., Gummadi, K. P., Gribble, S. D. A Measurement Study of Peer-to-Peer File Sharing Systems. In Proceedings of MMCN 2002.
12. Stutzbach, D., Zappala, D., Rejaie, R.: Swarming: Scalable Content Delivery for the Masses. University of Oregon Computer and Information Science Technical Report, UO-CIS-TR-2004-1, 2004.

13. Tewari, S., Kleinrock, L.: Analysis of Search and Replication in Unstructured Peer-to-Peer Networks, In Proceedings of ACM SIGMETRICS 2005.
14. Yang, X., de Veciana, G.: Service Capacity of Peer to Peer Networks. In Proceedings of ACM INFOCOM 2004.
15. Zou, L., Ammar, M.: A File-Centric Model for Peer-to-Peer File-Sharing Systems." In Proceedings of ICNP 2003.

A Novel Direct Upper Approximation for Workload Loss Ratio in General Buffered Systems

József J. Bíró, András Gulyás, and Zalán Heszberger

Budapest University of Technology and Economics,
Department of Telecommunications and Media Informatics,
1117 Budapest, Magyar tudósok körútja 2. Hungary
{gulyas, biro}@tmit.bme.hu

Abstract. The workload loss ratio (WLR) is a key quantity from the point of Quality of Service (QoS) provisioning in packet-based communication, hence it's estimation is an important issue. The existing results in the area of WLR approximation usually interpret the workload loss as a product of some well assessable quantities. We call this approach as the indirect approximation of the WLR. The drawback of this approach is that each estimation has an error and the product of these errors could result in a highly inaccurate bound. This work deals with the upper approximation of the workload loss ratio based on it's original definition and proposes a new direct bound on the WLR applicable in general service curve network element. Extensive and systematic performance analysis of the formulae have been done, in which we found that the direct approach gives more accurate bound in several cases. We illustrate this analysis through some numerical examples.

Keywords: Scheduling, resource estimation, statistical multiplexing.

1 Introduction

The increasing number of real-time Internet applications induce the preface of new services in telecommunication networks, besides best effort. These services have to meet some Quality of Service (QoS) requirements, which usually consist of prescriptions for QoS parameters. Thus the provision of QoS for packet switched networks generally means keeping the value of some quality related parameters at a level that fills these prescriptions. Since a significant part of the Internet applications are sensitive to the loss of the packets, the approximation of the workload loss ratio (WLR) parameter receives a significant attention. However the direct approximation of the prospective workload loss within network elements, multiplexing several inputs turns out to be a highly nontrivial problem. The probability of buffer overflow $Pr(Q > q)$ of an infinite queue is frequently used for WLR estimation [1] [2], nevertheless it is shown, that the ratio $\frac{WLR}{Pr(Q>q)}$ can be arbitrary [3]. Since the buffer overflow is closely related to

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 718–729, 2005.

the packet loss, it can be consumed for approximating the WLR indirectly as done in [3] [4] [5].

If the system is stationary and ergodic the following definition can be used[1]:

$$WLR = \frac{E[\text{number of lost bits}]}{E[\text{number of bits arriving}]} \tag{1}$$

We show, that estimating the packet loss in a direct manner using the definition (1) results in closed form bound which performs better than the existing WLR bounds. For the construction of the new bound only a little information is used about the input traffic (peak rate, upper estimated mean rate of the aggregate) so it might directly be applied in traffic management functions like call admission control (CAC) as well, without any complex measurement or information propagation.

Most of the existing bounds can be applied for constant rate servers only. We form our statements for more general queuing systems that can be described by a service curve property. The service curve property as defined in network calculus [6], with service curve β means, that at any time t, the observed output traffic in $[0, t]$ is at least equal to $A(s) + \beta(t - s)$ for some s in $[0, t]$, where $A(s)$ is the total input traffic in $[0, s]$. Using this definition, we derive new WLR formulae that can be used for a larger set of network elements, rather than for constant rate servers only.

Another problem of the existing closed form bounds for systems that satisfy the service curve property, that they assign different formulae for the case when the system is fed with inputs with the same characteristics (so called homogeneous case), and for the case when the properties on the inputs are different (heterogeneous case). We derive a universal bound that cover both cases.

Two different bounding approaches have been identified in [7] and [8]. The first one [7] based on the decomposition of the investigated network element, into virtual mini-nodes, that process one microflow as an input, and has a certain amount of processing capacity, usually a fraction of the entire server capacity. The summation of the lost packets in these mini-nodes gives an approximation of the lost packets within the original system. In the followings this approach is referenced as VNP (Virtual Node Partitioning). The other way to estimate the number of lost packets [8] will be named as Busy Period Partitioning (BPP), since it assigns a union bound for the lost packets on the time partition of the maximal possible busy period in which the packet loss could occur. Both of these approaches applied in [9] for buffer overflow and (indirect) workload loss ratio bounds in service curve network element. Because the BPP approach turned out to be more powerful in bounding buffer overflow as well as workload loss ratio [9], hereafter we concentrate on this approach.

In Section 2 we present different methods for the upper estimation of the probability generating function (PGF) used in bounding the WLR. Then in sec-

[1] Without loss of generality we consider fluid-like bit-processing system, since it can be shown, that the result can be applied for systems with rougher granularity (cells, packets).

tion 3.2 some existing results are presented in the area of indirect WLR approximation. Section 3.3 introduces a way to approximate the WLR in a definition based manner along the BPP methods, then we formalize our new bound. We illustrate through numerical examples that our direct formula performs better than the corresponding indirect one.

2 Theoretical Background

Buffer overflow and WLR approximation in service curve network element with regulated inputs [9] relies on bounding the tail distribution of sum of random variables with finite support.

One of the widely used approximation techniques for this tail distribution is the Chernoff-Hoeffding bounding method, which looks like as

$$P(X > q) \leq \inf_{\theta>0} \frac{G_X(\theta)}{e^{\theta q}} \leq \inf_{\theta>0} \frac{\tilde{G}_X(\theta)}{e^{\theta q}}, \tag{2}$$

where $G_X(\theta) = E[\exp(\theta X)]$ is the probability generating function (PGF) of X and $\tilde{G}_X(\theta)$ is a kind of reasonable bound of $G_X(\theta)$ (i. e. $G_X(\theta) \leq \tilde{G}_X(\theta)$). From (2) it can be seen, that giving a better bound on the PGF, gives a better bound on the buffer overflow as well.

The following two lemmas presents Hoeffding's results (using our notations) on the PGF approximation [10] for the homogeneous and heterogeneous cases.

Lemma 1 ([10]). *Let $X_1, X_2, ..., X_I$, independent random variables with $X = \sum_{i=1}^{I} X_i$, $M = E[X]$ and $0 \leq X_i \leq p$. Then for $\theta > 0$*

$$G_X(\theta) \leq \left(1 - \frac{M}{Ip} + \frac{M}{Ip} e^{(\theta p)}\right)^I. \tag{3}$$

Lemma 2 ([10]). *Let $X_1, X_2, ..., X_I$, independent random variables with $X = \sum_{i=1}^{I} X_i$, $M = E[X]$ and $0 \leq X_i \leq p_i$. Then for $\theta > 0$*

$$G_X(\theta) \leq e^{\theta M} e^{\frac{\theta^2 \sum_{i=1}^{I} p_i^2}{8}}. \tag{4}$$

One can see, that Lemma 2 does not coincide with Lemma 1 for the special setting of $p_1 = p_2 = ... = p_n$. The following PGF approximation leads to a formula that covers both cases.

Lemma 3 ([11],[12]). *Let $X_1, X_2, ..., X_I$ be I independent (and not necessarily identically distributed) random variables with $0 \leq X_i \leq p_i, X = \sum_{i=1}^{I} X_i$ and $M = E[X]$. Then for $s > 0$*

$$G_X(\theta) \leq \left(1 - \frac{M}{I^* p} + \frac{M}{I^* p} e^{\theta p}\right)^{I^*}, \tag{5}$$

where the right side is the PGF of the sum of $X_1^*, X_2^*, ..., X_{I^*}^*$, $I^* = \lceil \sum_{i=1}^{n} p_i/p \rceil$ *independent homogeneous Bernoulli random variables, with the identical peak value* $p = \max(p_1, p_2, ..., p_I)$ *and identical mean value* $E[X_i^*] = M/I^*$*.*

Although this PGF bound is suitable to cover both the homogeneous and heterogeneous cases in bounding the buffer overflow probability, we need further results for obtaining uniform WLR bound. For the relation of random variables X and X^* a much more general results is valid (with the apparent combination of Lemma 1 and Theorem 3 in [13] and also found in [14]), namely

$$\frac{E[(X-q)^+]}{E[X]} \le \frac{E[(X^*-q)^+]}{E[X^*]} \ ,\forall q \ge 0 \tag{6}$$

where $X^* = \sum_{i=1}^{I^*} X_i^*$. Note that the random variable X^* has binomial distribution, hence,

$$\frac{E[(X^*-q)^+]}{E[X^*]} = \frac{p}{M} \sum_{l=\lceil q \rceil}^{I^*} (l - \lceil q \rceil) \binom{I^*}{l} \left(\frac{M}{I^* p}\right)^l \left(1 - \frac{M}{I^* p}\right)^{I^*-l} \ . \tag{7}$$

It can be shown by straightforward calculation that the derivative of this function with respect to M is always non-negative, that is

$$\partial_M \frac{E[(X^*-q)^+]}{E[X^*]} \ge 0 \ , \ \forall q > 0 \ . \tag{8}$$

A very practical consequence of this result is that if only an upper bound of M is known, then the inequality still holds in (6) when this known upper bound of M is used in the computation or an upper approximation of the right hand side of (6).

3 Upper Approximations of WLR

3.1 Notation and Assumptions

In this paper the following notations of network calculus [6] are used: $\mathcal{I} = \{1, 2, ..., I\}$ is a set of input flows in a network element. $A_i(s,t], i \in \mathcal{I}$ denotes the number of bits arrived in input i in the interval (s,t]. $A_i^*(s,t], i \in \mathcal{I}$ means the same for the output of the ith flow. Let $A(s,t] := \sum_{i=1}^{I} A_i(s,t]$, and $A^*(s,t] := \sum_{i=1}^{I} A_i^*(s,t]$. The notation $v(f,g) = \sup_{t \ge 0}\{f(t) - g(t)\}$ stands for the maximal vertical, and the notation $h(f,g) = \sup_{t \ge 0}\{\inf\{u \ge 0 : f(t) \le g(t+u)\}\}$ for the maximal horizontal deviation between f and g. We define $\bar{\alpha} = \sum_{i=0}^{I} \bar{\alpha}_i$, where $\lim_{u\to\infty} A_i(0,u]/u \le \lim_{u\to\infty} \alpha_i(u)/u = \bar{\alpha}_i$, and $\alpha = \sum_{i=0}^{I} \alpha_i$.

The following assumptions are also needed for the derivation.

- $(A1)$ $A_1, A_2, ..., A_I$-s are independent
- $(A2)$ For all $i \in \mathcal{I}, A_i$ has α_i as an arrival curve, where α_i is a non-negative wide-sense increasing function.

- $(A3)$ For each $i \in \mathcal{I}$, and any $s, t \in R$, $E[A_i(s,t]] \leq \bar{\alpha_i} \cdot (t-s)$, where $\bar{\alpha_i} = \lim_{t \to \infty} \alpha_i(t)/t$.
- $(A4)$ There exists a sequence of random points: $... < S_{-2} < S_{-1} < S_0 \leq 0 < S_1 < S_2 < ...$, such that $\lim_{n \to -\infty} S_n = -\infty$, and $\lim_{n \to \infty} S_n = \infty$, and for all $n \in Z, A(S_n, S_{n+1}] = A^*(S_n, S_{n+1}]$
- $(A5)$ If $S(t) = \{S_n, n \in Z : S_n \leq t\}$, and β is the aggregate service curve for the flows, than for all $t \in R$, and any $u \in S(t)$, $\exists s \in [u,t] : A^*(u,t] - A(u,s] \geq \beta(t-s)$, where β is a non-negative wide sense increasing function.
- $(A6)$ There exists[2] $\tau < \infty$ such that for all $s \geq \tau$, $\beta(s) \geq \alpha(s)$.
- $(A7)$ Let A_i and A_i^* be stationary and ergodic.

3.2 Indirect Bounds on the WLR

In our explanation the indirect derivation in the approximation of the WLR means, that the given method does not estimate quantity that defines the workload loss ratio, but interprets it as a product of other quantities, and defines upper bounds on each of these related quantities. For systems that satisfy the service curve property such bound is proposed in [4], where the WLR estimator formula is the product of the bound on the buffer overflow probability and an additional term, which is a hard deterministic bound on the loss ratio over any time interval of length t. The following Theorem recalls that result.

Theorem 1 ([9]). Assume $(A1) - (A3)$, and that $v(\alpha, \beta) < \infty$, $h(\alpha, \beta) < \infty$, and $\alpha_i = \alpha_1$ for all $i \in I$. Then

$$WLR \leq \frac{\hat{l}(1)\alpha(1)}{\bar{\alpha}} P(Q^\infty(0) > q) \tag{9}$$

where $\hat{l}(t) = 1 - \inf_{s \leq t} \frac{\beta(s)+q}{\alpha(s)}$, and $Q^\infty(t)$ is the buffer occupancy of a virtual system identical to the original system, but with a buffer size sufficient to ensure no losses.

Theorem 1 defines a framework for WLR approximation in an indirect manner. The substitution of the existing buffer overflow bounds for service curve network elements according to the VNP and the BPP methods results in different formulae for the WLR estimation. Since these result need a buffer overflow estimation, the bound inherits the undesirable property, that the buffer overflow bounds presented in [4] splits into two different formulae, for the homogeneous and the heterogeneous cases. The reason for this is the use of two different PGF approximation ((3), (4)) for these two cases and as a corollary two bounds raise for each bounding approach (VNP, BPP). The presentation of these formulae is omitted here, one can easily recover them from [9] and [4]. Among these bounds in this paper we use the indirect BPP-based WLR bound assuming heterogeneous inputs as a reference and for comparison purposes to our direct BPP-based WLR bound.

[2] The maximum possible busy period in such a system is τ.

3.3 WLR Estimation with Direct Formula

According to the definition of the WLR for stationary and ergodic systems [3],[15]:

$$WLR = \frac{E[\text{number of lost bits in a unit time interval}]}{E[\text{number of bits arriving in a unit time interval}]} \tag{10}$$

The expected value of the number of lost bits in a finite buffer system, can be bounded from above by the number of packets overflown in the infinite buffer system [3]:

$$WLR \leq \frac{E[(Q^{\infty} - q)^{+}]}{E[A]}$$

where Q^{∞} represents the stationary buffer occupancy of the system with infinite buffer, and $E[A] = E[A(0,1)]$ is the number of bits arriving in a unit time interval.

3.4 Bounds Using He BPP Approach

In what follows we derive bound on the WLR according to the BPP approach. To this end the following two lemmas are needed.

Lemma 4 ([9]). *If the assumptions (A2),(A5) hold and furthermore β is superadditive then*

$$Q(0) \leq \sup_{0 \leq s \leq \tau} \{(A(-s,0) - \beta(s))\} \tag{11}$$

where $Q(0)$ represents the buffer occupancy at an arbitrary time instant 0.

According to assumption (A7) Lemma 4 is also valid for the stationary buffer occupancy Q^{∞}, hence

$$WLR \leq \frac{E[\sup_{0 \leq s \leq \tau}\{(A(-s,0) - \beta(s)\} - q)^{+}]}{E[A]}. \tag{12}$$

Before the presentation of our WLR bound based on equation (12), another lemma has to be considered. For any $K \in N$, and any $t \geq 0$, let $T_K(t)$ be the set of partitions of $[0,t)$ in K intervals: $T_K = \{(t_0, t_1, ..., t_K) : 0 = t_0 \leq t_1 \leq ... \leq t_K = t\}$.

Lemma 5 ([9]). *For any $K \in N$, $\boldsymbol{t} \in T_K(\tau)$ and $q \geq 0$,*

$$Pr[\sup_{0 < s \leq \tau} \{(A(-s,0) - \beta(s)\} > q] \leq \sum_{k=0}^{K-1} Pr[(A(0,t_{k+1}) > q + \beta(t_k))] \tag{13}$$

The next theorem gives a direct approximation of the WLR, based on inequality (12):

Theorem 2. Assume $(A1) - (A7)$. Then

$$WLR \leq \inf_{K \in N, \boldsymbol{t} \in T_K(\tau)} \frac{1}{\bar{\alpha}} \sum_{k=0}^{K-1} \\ \theta_k^* \left(\frac{m_k}{c_k}\right)^{\frac{c_k}{\hat{\alpha}_{t_{k+1}}}} \left(\frac{I_{k+1}^* \hat{\alpha}_{t_{k+1}} - m_k}{I_{k+1}^* \hat{\alpha}_{t_{k+1}} - c_k}\right)^{I_{k+1}^* - \frac{c_k}{\hat{\alpha}_{t_{k+1}}}} \tag{14}$$

where $m_k = \bar{\alpha} t_{k+1}$, $c_k = \beta(t_k) + q$, $I_{k+1}^* = \left\lceil \frac{\sum_{i=1}^{I} \alpha_i(t_{k+1})}{\hat{\alpha}_{t_{k+1}}} \right\rceil$, $\hat{\alpha}_{t_{k+1}} = \max_{i \in I}(\alpha_i$ $(t_{k+1}))$, and $\theta_k^* = \frac{1}{\hat{\alpha}_{t_{k+1}}} \log \frac{c_k}{m_k} \frac{I_{k+1}^* \hat{\alpha}_{t_{k+1}} - m_k}{I_{k+1}^* \hat{\alpha}_{t_{k+1}} - c_k}$.

Proof. Using a well-known computation of the expected value of non-negative random variable:

$$\frac{E[\sup_{0 \leq s \leq \tau}\{(A(-s,0) - \beta(s)\} - q)^+]}{E[A]} = \\ = \frac{\int_{x=0}^{\infty} Pr[\sup_{0 \leq s \leq \tau}\{(A(-s,0) - \beta(s)\} > q + x]dx}{E[A]}$$

In accordance with Lemma 5:

$$\frac{\int_{x=0}^{\infty} Pr[\sup_{0 \leq s \leq \tau}\{(A(-s,0) - \beta(s)\} > q + x]dx}{E[A]} \leq \\ \leq \frac{\int_{x=0}^{\infty} \sum_{k=0}^{K-1} Pr[A(0,t_{k+1}) > q + \beta(t_k) + x]dx}{E[A]}.$$

By the commutation of the integration and summation we get:

$$\frac{\int_{x=0}^{\infty} \sum_{k=0}^{K-1} Pr[A(0,t_{k+1}) > q + \beta(t_k) + x]dx}{E[A]} \leq \\ \leq \frac{\sum_{k=0}^{K-1} \int_{x=0}^{\infty} Pr[A(0,t_{k+1}) > q + \beta(t_k) + x]dx}{E[A]}.$$

One element in the summation on the right hand side can be written as

$$\int_{x=0}^{\infty} Pr[A(0,t_{k+1}) > q + \beta(t_k) + x]dx = E[(A(0,t_{k+1}) - q - \beta(t_k))^+] \tag{15}$$

Due to the assumed stationary increment of $A(0,t)$, $E[A]$ can be rewritten as

$$E[A] = \frac{E[A(0,t_{k+1})]}{t_{k+1}}\ ,\ k = 0, \ldots, K-1\ . \tag{16}$$

Using this it follows that

$$WLR \leq \sum_{k=0}^{K-1} \frac{E[(A(0,t_{k+1}) - q - \beta(t_k))^+]}{E[A(0,t_k+1)]} t_{k+1} \tag{17}$$

Since $A(0,t_{k+1}) = \sum_{i=1}^{I} A_i(0,t_{k+1})$, $A_i(0,t_{k+1}) \le \alpha_i(t_{k+1})$ and $E[A_i(0,t_{k+1}] \le \bar{\alpha}_i t_{k+1}$ we can apply the results presented in Section 2. That is let $A_i^*(0,t_{k+1})$, $i = 1,\ldots,I^*$ independent and identically distributed Bernoulli random variables with mean $E[A_i^*(0,t_{k+1})] = \bar{\alpha} t_{k+1}/I^*$, $A_i^*(0,t_{k+1}) \le \hat{\alpha}_{t_{k+1}}$, where

$$\hat{\alpha}_{t_{k+1}} = \max_{1\le i\le I}(\alpha_i(t_{k+1}))\ ,\ I^* = \left\lceil \frac{\sum_{i=1}^{I} \alpha_i(t_{k+1})}{\hat{\alpha}_{t_{k+1}}} \right\rceil \tag{18}$$

According to (6) and (8) one can deduce that

$$\frac{E[(A(0,t_{k+1}) - q - \beta(t_k))^+]}{E[A(0,t_k+1)]} t_{k+1} \le \frac{E[(A^*(0,t_{k+1}) - q - \beta(t_k))^+]}{\bar{\alpha}} \tag{19}$$

where $A^*(0,t_{k+1}) = \sum_{i=1}^{I^*} A_i^*(0,t_{k+1})$ and hence $E[A^*(0,t_{k+1})] = \bar{\alpha} t_{k+1}$. Applying a well-known Chernoff bound [16] to the right hand side of (19) gives

$$E[(A^*(0,t_{k+1}) - q - \beta(t_k))^+] \le \frac{1}{\theta_k^*} \frac{E\left[\exp(\theta_k^* A^*(0,t_{k+1}))\right]}{\exp(\theta_k^*(q+\beta(t_k)))} \tag{20}$$

where[3]

$$\theta_k^* = \mathrm{arginf}_{\theta_k}\big(\log E\left[\exp(\theta_k A^*(0,t_{k+1}))\right] - \theta_k(q+\beta(t_k))\big) \tag{21}$$

The generating function of $A^*(0,t_{k+1})$ is

$$E[\exp(\theta_k A^*(0,t_{k+1}))] = \left(1 - \frac{\bar{\alpha} t_{k+1}}{I^* \hat{\alpha}_{t_{k+1}}} + \frac{\bar{\alpha} t_{k+1}}{I^* \hat{\alpha}_{t_{k+1}}} \exp(\theta_k \hat{\alpha}_{t_{k+1}})\right)^{I^*} \tag{22}$$

Substituting it into the Chernoff bound in (20) and after straightforward calculations the closed form formula of (14) is obtained. Q.E.D.

4 Evaluation

For illustrating the evaluation and comparison of the bounds presented the following scenario is used. We have 100 input flows, which are token bucket constrained with some arrival curve ($\alpha(t) = \bar{\alpha} + \sigma$), and the packet forwarder satisfies a rate latency service curve property, with $\beta = c \cdot max(t-e, 0)$, in a work-conserving manner. We take parameter values that are close to many practical, common applications. The service rate of the server will be 150Mbps and let the packets size be 1500 bytes. This means the node can serve 12500 packets during a second (pps). The latency time (e) is $8 \cdot 10^{-5}$ sec. We set up four configurations for the evaluation:

[3] Note that θ_k^* does not provide the optimal Chernoff bound, however, it guarantees closed form WLR bound as opposed to the optimal $\mathrm{arginf}_{\theta_k}\big(\log E\left[\exp(\theta_k A^*(0,t_{k+1}))\right] - \theta_k(q+\beta(t_k) - \log\theta_k)\big)$.

Table 1. The summary of configurations

Conf.	$\alpha_1(t)$	$\alpha_2(t)$	$\alpha(t)$	utilization
Conf. 1.	$33.3pps + 8p$	$16.6pps + 5p$	$2500pps + 650p$	0.2
Conf. 2.	$133.3pps + 8p$	$66.6pps + 5p$	$10000pps + 650p$	0.8
Conf. 3.	$26pps + 8p$	$24pps + 8p$	$2500pps + 800p$	0.2
Conf. 4.	$102pps + 8p$	$88pps + 8p$	$10000pps + 800p$	0.8

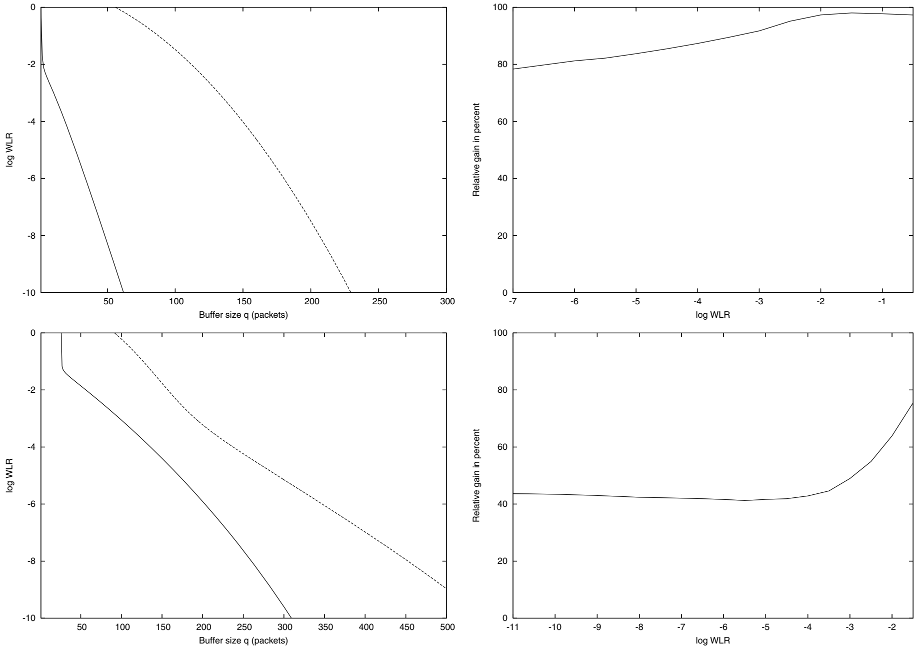


Fig. 1. The results of Theorem 1 and Theorem 2 for conf. 1 (up) and conf. 2 (down)

Configuration 1: $\alpha_1(t) = 33.3pps + 8p$, $\alpha_2(t) = 16.6pps + 5p$. If we have 50 microflows with $\alpha_1(t)$ and $\alpha_2(t)$ each, it results $\alpha(t) = 50 \cdot \alpha_1(t) + 50 \cdot \alpha_2(t) = 2500pps + 650p$ as an aggregate arrival curve. This configuration represent a utilization of 0.2 for the server.

Configuration 2: $\alpha_1(t) = 133.3pps + 8p$, $\alpha_2(t) = 66.6pps + 5p$. If we have 50 microflows with $\alpha_1(t)$ and $\alpha_2(t)$ each, it results $\alpha(t) = 50 \cdot \alpha_1(t) + 50 \cdot \alpha_2(t) = 10000pps + 650p$ as an aggregate arrival curve. This configuration represent a utilization of 0.8 for the server.

Configuration 3: $\alpha_1(t) = 26pps+8p$, $\alpha_2(t) = 24pps+8p$. 100 microflows results in $\alpha(t) = 2500pps+800p$ as an aggregate arrival curve. This configuration represent a utilization of 0.2 for the server.

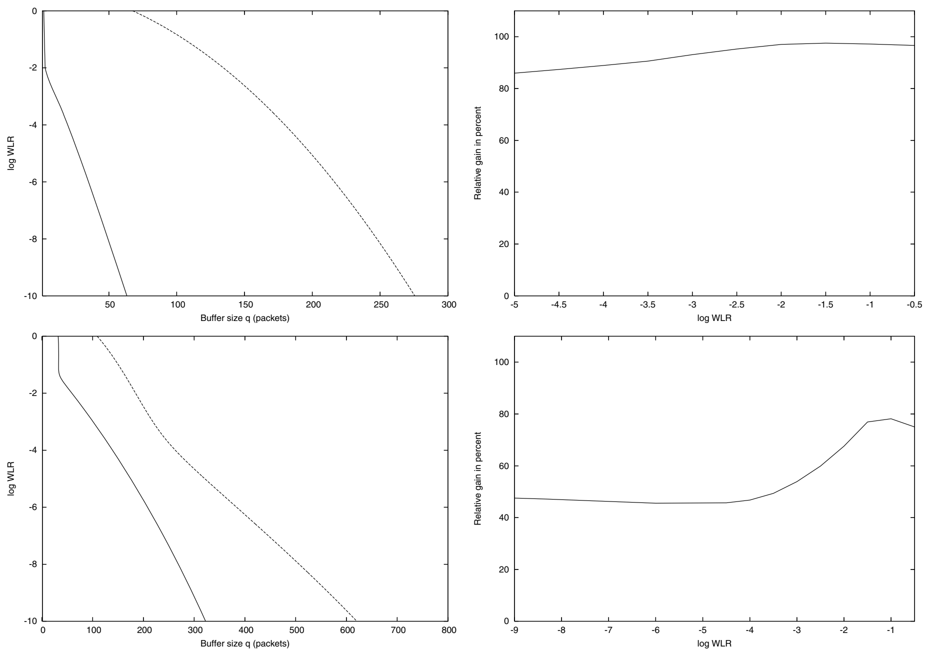


Fig. 2. The results of Theorem 1 and Theorem 2 for conf. 3 (up) and conf. 4 (down)

Configuration 4: $\alpha_1(t) = 102pps + 8p$, $\alpha_2(t) = 88pps + 8p$. 100 microflows results in $\alpha(t) = 10000pps + 800p$ as an aggregate arrival curve. This configuration represent a utilization of 0.8 for the server. A summary of the configurations can be seen on Table 1.

The choice of configuration 3 and 4 is to evaluate the performance of the bounds, when the inputs have almost same characteristics. Therefore the scenario is close to the homogeneous case, but since it is heterogeneous the heterogeneous form of the PGF approximation have to be used.

For each configuration we have two graphs. The one on the left side shows the logWLR approximation as a function of the buffer size (the continuous line represents our new bound, while the dotted line represents the old bound) and the graph to the right indicates the relative buffer requirement gain according to the two bounds. This latter quantity refers to the amount of bandwidth which can be saved when our new bound is used to keep the loss ratio under a certain level. This is defined in the following way. Let $Q_{\mathrm{req},1}$ and $Q_{\mathrm{req},2}$ be the buffer requirements formulated as

$$Q_{\mathrm{req},1}(-\gamma) = \min(q,\, WLR_1(q) \le 10^{-\gamma}) \;,\; Q_{\mathrm{req},2}(-\gamma) = \min(q,\, WLR_2(q) \le 10^{-\gamma}) \tag{23}$$

where $WLR_1(q)$ and $WLR_2(q)$ are the WLR approximations according to Theorem 1 and Theorem 2 . The relative gain in buffer requirement drawn in the graphs is computed as

$$\text{gain}(-\gamma) = \frac{Q_{\text{req},1}(-\gamma) - Q_{\text{req},2}(-\gamma)}{Q_{\text{req},1}(-\gamma)} \tag{24}$$

All figures well illustrate our experience in extensive comparison that our new bound significantly outperforms the old WLR approximation, especially in case of low traffic intensity. For higher traffic loads the relative buffer requirement gain decreases, but still remains high, around 40-50% in the examples. The relative gain is usually higher for less stringent QoS constraints on the WLR (smaller γ). In Fig. 1 for conf. 1 one can observe that to achieve no more than 10^{-6} loss ratio the required buffer size is smaller than $50p$ when WLR_2 is used and it is around $180p$ when WLR_1.

5 Conclusions

The scope of this paper was to present our new direct (definition based) formula for the workload loss ratio applicable in general buffered systems characterized as service curve network element. Our new bound is significantly better then the corresponding existing one, and can ensure to save tremendous amount of buffer space when it is used to guarantee QoS level for the workload loss ratio.

References

1. N. G. Duffield, J. T. Lewis, N. O'Connel, R. Russel, and F. Foomey. Entropy of atm traffic streams: tool for estimating qos parameters. *IEEE Journal of Selected Areas in Communications*, 13, March 1995.
2. M. Krunz and A. M. Ramasamy. The correlation structure for a class of scene-based video models and its impact on the dimensioning of video buffers. *IEEE Trans. Multimedia*, 2, July 2000.
3. András György and Tamás Borsos. Estimates on the packet loss ration via queue tail probabilities. *IEEE Globecom*, March 2001.
4. Milan Vojnovic and Jean-Yves Le Boudec. Stochastic analysis of some expedited forwarding networks. *IEEE INFOCOM New York*, June 2002.
5. Nikolay Likhanov and Ravi R. Mazumdar. Cell loss asymptotics in buffers fed with a large number of independent stationary sources. *Journal of Applied Probability*, 36, March 1999.
6. Jean-Yves Le Boudec and Patrick Thiran. Network calculus, 2002.
7. George Kesidis and Takis Konstantopoulos. Worst-case performance of a buffer with independent shaped arrival processes. *IEEE Communication Letters*.
8. C.-S. Chang, W. Song, and Y. Ming Chiu. On the performance of multiplexing independent regulated inputs. *Proceedings of Sigmetrics*, May 2001.
9. Milan Vojnovic and Jean-Yves Le Boudec. Bounds for independent regulated inputs multiplexed in a service curve network element. *IEEE Trans. on Communications*, 51(5):449–451, May 2003.

10. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58:13-30*, March 1963.
11. J. Bíró, A. Gulyás, and M. Martinecz. Parsimonious estimates of bandwidth requirement in quality of service packet networks. *Performance Evaluation*, 59(2–3):159–178, February 2005.
12. J. Bíró, Z. Heszberger, and M. Martinecz. A family of performance bounds for qos measures in packet-based networks. In *IFIP Networking 2004*, pages 1108–1119, 2004.
13. G. Mao and D. Habibi. Loss performance analysis for heterogeneous on-off sources with application to connection admission control. *IEEE Transactions on Networking*, July 2001.
14. R. Szekli. *Stochastic ordering and dependence in applied probability (Lecture Notes in Statistics).*
15. M. Vojnovic and J. Y. Le Boudec. Stochastic analysis of some expedited forwarding networks. Technical Report DSC/2001/039, EPFL-DI-ICA, July 2001.
16. F. P. Kelly. Notes on effective bandwidth. *Stochastic Networks: Theory and Applications*, 4, Sep 1995.

PISA: Automatic Extraction of Traffic Signatures

Parminder Chhabra[1], Ajita John[2], and Huzur Saran[3]

[1] Winlab, Rutgers, The State University of New Jersey, NJ, USA
pchhabra@alumni. utexas.net
[2] Avaya Labs Research, Lincroft, NJ, USA
ajita@avaya.com
[3] Dept. of CS & Eng., Indian Institute of Technology, New Delhi, India
saran@cse.iitd.ernet.in

Abstract. Analysis of security attacks shows that an attack leaves its imprint or signature in the attack packets. Traffic from Distributed Denial of Service attacks and rapid worm spreads has the potential to yield signatures. While all signatures may not be indicative of attacks, it is useful to extract non-transient signatures that are carried by a sufficient number of flows/packets/bytes. The number of packets/bytes in the flows carrying the signature may be used for rate-limiting the flows, providing for timely and automated response to both known and unknown attacks. This paper proposes an efficient algorithm, PISA, which clusters flows based on similarity in packet information and extracts signatures from high-bandwidth clusters. Extensive experiments on two weeks of real attack data of 100 million packets yield about 1744 signatures. Additionally, PISA extracted the signature for the Blaster worm connection attempts in a mix of traffic from a trans-Pacific backbone link.

Keywords. Signatures, Traffic Clusters, Security, DDoS, Worms.

1 Introduction

Worms such as Code Red and Slammer [1] are spread by the repeated execution of similar pieces of malicious code. Distributed Denial of Service (DDoS) attacks are also typically caused by similar pieces of code executing on compromised hosts and launching network flows. The code exploits some vulnerability in the OS, an application or parts of the TCP/IP protocol and causes network systems to experience degradation in performance. When the same program is responsible for an attack, packets belonging to different flows of the attack may share the same values for several fields in the packets such as protocol header fields, application level fields and packet content. For example, in a SYN attack, the values for packet size, protocol, the SYN flag in the TCP header, destination address (if a particular server is being targeted), and destination port (if a particular kind of server is being targeted) are the same for all flows participating in the attack. The common values in the Code Red worm attack were in the following fields: *{message size, protocol, destination port}* [1]. Typically, unsolicited response traffic has common values for packet size and error type / code in ICMP packets and TCP flags in TCP packets. Some attacks launched by commonly used attack tools [7] have the same header checksum and

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 730 – 742, 2005.

associated IP options. Additionally, similarity in application-level fields such as Subject, Body, and Attachment filenames can be found in SMTP-based email viruses.

Schemes that investigate packet header fields to protect against security attacks include [6], [9] and [3]. In [6], the authors propose a technique called Hop-Count Filtering to cluster destination address prefixes based on hop count values. Departures from baseline values of the TTL field in the IP header (hop count) are used to identify spoofed source addresses. A scheme for using RED-drop history as a way of identifying singleton high-bandwidth flows and aggregating these flows based on prefixes of destination IP addresses is proposed in [9]. Specialized techniques to aggregate flows based on specific fields (source address and port, destination address and port, and protocol) are proposed in [3]. Schemes that investigate packet content include [8] and [12]. In [8], the authors use a port-scan classifier to first identify suspected malicious flows. Next, they use string matches on packet content using Rabin fingerprints together with protocol and destination port as a way of identifying worm signatures. In [12], the authors use string matches on packet content together with destination port as a basis for identifying worm signatures. The above schemes are effective for the specific fields they target. However, there exists a need to study the problem of aggregating traffic using any subset of fields because, as known attacks have shown and unknown attacks may possibly show, characteristics of attacks may extend to other fields. Investigating any subset of fields may lead to the automatic generation of generalized attack profiles that may form the basis for an automated response to unknown attacks. There are, of course, varying degrees of cost for retrieving values for different fields in a packet. However, as the severity of attacks continue to increase, this cost may be justified.

The work described in this paper addresses the problem of finding high bandwidth aggregates of flows that share similar values for any subset of fields in their packets. This subset of fields and their values form *signatures*. While all signatures may not be indicative of attacks, signatures that are carried by a large number of flows/packets/bytes and occur frequently in traffic are of high interest to network administrators for early detection of possible malicious activity. The paper discusses packet signatures and their relevant properties. The paper discusses the Packet Imprint in Security Attacks algorithm (PISA). PISA samples incoming packets at a network element and groups the flows corresponding to the packets into clusters based on similar values in the fields of the packets. A cluster is intensive if it consumes a large proportion of the bandwidth as measured by the number of packets/bytes across all flows in the cluster. PISA extracts the signatures of intensive clusters from samples of data over time and filters out transient signatures. The signatures carry associated information such as the number of samples that the signatures appeared in (persistence) and the average number of flows (distribution) and packets/bytes (intensity) that belong to the signature. PISA provides an aggregated view of the traffic that arrives at a network element and the aggregation is not constrained to any particular field in the packet. The effectiveness of our approach is shown through experiments with an implementation of PISA on real network data, which demonstrate that PISA can extract signatures of high value such as those that correspond to connection attempts in a worm spread. Additionally, PISA was able to aggregate large amounts of real attack data to yield signatures that showed distribution of the attacks over a variety of fields.

The key contributions of our work are as follows: We define characteristic properties of signatures – Dimension, Intensity, Persistence and Distribution. We propose PISA, a randomized non-exponential algorithm for clustering traffic flows based on any subset of fields in packets to yield signatures. Signatures may carry any combination of fields that may be defined for a packet. No pre-computed signatures are required. Only a sample of traffic is required to extract significant signatures. In the face of extreme performance degradation at a network element, PISA can be used as a basis for self-healing of the network element by penalizing traffic that consumes the most resource(s).

The paper is organized as follows. Section 2 discusses signatures and clusters of flows. Section 3 describes PISA. Section 4 discusses experiments and results on traffic traces. Finally, conclusions are presented in Section 5.

2 Signatures from Clusters of Network Flows

Definition: A signature is given by k ordered pairs $\{(f_1, v_1), (f_2, v_2), \ldots (f_k, v_k)\}$, where $k \geq 1$ is the dimension of the signature, f_i is a field in a packet and v_i is a value for f_i, $1 \leq i \leq k$.

A signature S is a subset of another signature T if the ordered pairs in S are contained in T. A flow is said to *carry* a signature if the ordered pairs in the signature match with values for the corresponding fields in the packets of the flow. The notion of a flow used in this work is a 5-tuple with the following fields: (source address, source port, destination address, destination port, protocol). An example of a signature is as follows: *{(packet_size, 48), (src_port, 80), (src_addr, 10.0.0.1), (type, tcp), (tcp_flag, Syn ack), (dest_port, 3072)}*. The signature is carried by flows originating from source 10.0.0.1 and port 80 with SYN ack packets of size 48 bytes to port 3072 of destinations. Four properties that characterize signatures are defined as follows:

1. Dimension: The dimension of a signature is the number of field-value pairs in it. A higher dimensional signature contains more information about the type of flows carrying the signature. A signature may be useful only if it contains a minimum number of fields in it.

2. Intensity: The intensity of a signature is the average number of packets/bytes that carry the signature. Intensity reflects the bandwidth consumption of flows carrying the signature.

3. Persistence: The persistence of a signature is its activity over time. It may be represented as a history of the frequency of occurrence of a signature over a time interval.

4. Distribution: The distribution of a signature is the average number of flows (alternatively, prefix matches of IP addresses across flows) carrying the signature and is useful in determining the spread of the signature.

The *significance* of a signature can be defined in terms of its dimensionality, intensity, persistence, and distribution. For example, a system may define significant signatures to be those that contain at least 4 fields, have a bandwidth consumption of at least 2% (intensive), appear for at least 60 seconds (persistent), and be carried by 5% of the traffic flows.

Definition: An *m*-dimensional cluster C_m is defined as a set of flows in which all flows have similar values for each field in a set of *m* fields. While exact matches are discussed in this paper as the basis of similarity, the approach can be extended to include approximate matches. The *m* fields form a signature carried by all flows in the cluster.

The set of flows in a network sample can be grouped into *clusters* of flows such that all flows in a cluster carry a "maximal" signature, where the maximal signature for a set of flows is the largest set of similar field-value pairs. A flow can belong to more than one cluster. Note the following properties:

- An m-dimensional cluster is also an n-dimensional cluster where $1 \le n \le m$.
- For two clusters X,Y with signatures S_x, S_y, $X \subset Y$, ➔ $S_y \subseteq S_x$ …….**(a)**

Each flow in a cluster has a weight given by the number of packets/bytes in that flow. The weight of a cluster can be defined as the sum total of the weights of all the flows in that cluster as follows: Weight(cluster C) = Sum of weights of all flows in C ….. **(b)**

The weight of a cluster is indicative of the bandwidth it consumes. An *intensive* cluster is one whose weight is above a specified threshold. *Intensive clusters yield intensive signatures.*

3 PISA: Algorithm

There has been prior work that has looked at identifying singleton flows that are intensive [9] [15]. The motivation in this work is to efficiently extract high-dimensional, intensive, persistent and distributed signatures in network traffic. The approach is to scan a sample of the incoming traffic at a network element and group the flows in that sample into intensive clusters, which yield intensive signatures. Different samples of data (closely spaced in time) yield a history of intensive signatures. Transient signatures can be pruned out from this history. The non-transient signatures along with related information such as the number of scans in which the signature was seen (persistence), the average number of flows (distribution) and the average number of packets/bytes (intensity) carrying each signature provides a history of aggregated traffic at the network element.

Clustering techniques can be broadly divided into hierarchical and partitional [2] [5]. Hierarchical clustering techniques consist of successively combining smaller clusters into larger ones or by splitting larger clusters into smaller ones. Partitional clustering techniques decompose the data into a set of disjoint clusters based on the optimization of a criterion function. Exhaustive techniques to search for all clusters in a data sample using any of the techniques discussed above are expensive [10]. We argue that exhaustive techniques are not necessary in this work because persistent and distributed signatures should be easily extracted in approximation-based methods. PISA repeatedly executes over samples of data across time intervals. Hence, it is not necessary to find all intensive signatures in an execution, as persistent signatures that are missed in one execution will likely show up in subsequent ones. Randomized approaches [4] allow faster extraction of clusters than exhaustive approaches. PISA uses a hierarchical and randomized clustering technique on a lattice structure (which is commonly used in clustering methods) of the sampled flows.

The parameters in PISA are as follows: (i) Sample size or the number of packets sampled in each scan phase: *N*. (ii) The minimum desired dimension of a signature: *k* (iii) Threshold for an intensive cluster: *T*. (iv) The fields of interest in the sampled packets. PISA consists of two parts: (1) A scan phase that extracts a sample of the network traffic (2) A signature extraction phase that clusters the flows in the sample to generate signatures.

In the scan phase, PISA samples a specified number of packets from the packets arriving at a network element. It maps the sampled packets to network flows and populates a *flow table* where each entry contains a flow, the number of packets belonging to the flow and the fields of interest in the packets and their corresponding values.

In the signature extraction phase, the flows in the flow table are grouped into clusters. Let *F* be the set of flows in the flow table. Consider a lattice of all possible subsets of *F* and referred to as the flow lattice in this paper. The empty set ϕ is at the bottom of the lattice and *F* is at the top of the lattice. An edge exists between two subsets S_1 and S_2 if S_1 is a superset of S_2 and $|S_1| = |S_2| + 1$. The lattice consists of $|F|+1$ levels where each level *i* has all the subsets of *F* with cardinality *i* (See Fig.1). The intensive clusters of *F* are points on the lattice. The goal is to find intensive signatures of dimension greater than or equal to *k*.

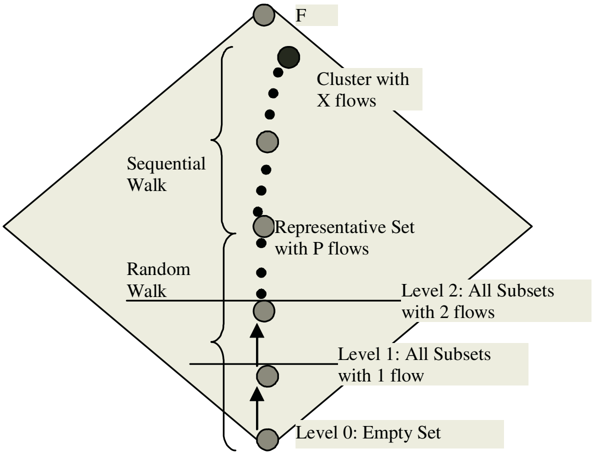

F
Cluster with X flows
Sequential Walk
Representative Set with P flows
Random Walk
Level 2: All Subsets with 2 flows
Level 1: All Subsets with 1 flow
Level 0: Empty Set

Fig.1. Lattice of Subsets of Flows and Signature Extraction

Let *h* be the number of fields of interest in the packets and k $(1 \leq k \leq h)$ be the minimum number of fields desired to be similar in a cluster. Consider a path $(\phi, S_1, S_2 \ldots, F)$ from the bottom of the lattice to its top. Trivially, ϕ and S_1 (consisting of a single flow) are *h*-dimensional clusters. Also, if *S* is a superset of *P* and *S* is an *i*-dimensional cluster and *P* is a *j*-dimensional cluster, then $i \leq j$ (follows from property (a) in Section 2). Thus, the dimension of a cluster (and the dimension of its associated signature) monotonically decreases up the lattice. This implies that for any $1 \leq k \leq h$, either F is a k-dimensional cluster or there must exist two subsets of F, S_t, S_{t+1}, such that S_t is an i-dimensional cluster, S_{t+1} is a j-dimensional cluster, there is an edge between S_t, and S_{t+1}, and $j < k \leq i$. S_t is called a k-dimensional representative set. Along the path from the bottom of the lattice to S_t, there may exist several representative sets of dimensions greater than k. The number of paths to a representative set is exponential in the number of flows in the representative set.

The signature extraction phase in PISA attempts to find a path to a representative set by randomly picking flows to move up the levels of the flow lattice. It starts at the bottom of the lattice by initializing a set S as the empty set ϕ. It traverses up the lattice in a sequence of steps. In each step it randomly chooses a flow from the flow table to add to the set S, each time checking if the resulting set is a representative set of dimension k or greater. It collects all such representative sets. It stops the traversal when the resultant set is a cluster of dimension less than k. This part of the walk will be called the random walk. A cluster forms the top of a sub-lattice that contains all subsets of the cluster. The more flows there are in the cluster, the larger is this sub-lattice and the greater the probability of remaining within the sub-lattice during the random walk and finding a representative set for the cluster.

At the end of the random walk, each collected representative set is expanded into a cluster by matching the headers of each flow in the flow table against those of the representative set. This corresponds to walking the sub-lattice corresponding to the cluster by sequentially searching the flow table. This part of the walk is called the sequential walk. The set resulting from the growth of the representative set is a cluster of flows that share common values for at least k fields. This cluster will yield an intensive signature if the weight of the cluster is greater than the specified threshold T.

Each walk through the lattice (random + sequential) will be referred to as a lattice traversal. Lattice traversals can be repeatedly executed to discover an increasing number of the intensive clusters in the lattice. The limit on the number of lattice traversals can be set through a parameter as discussed later. Note that PISA does not construct the entire lattice. It constructs sets of flows as it encounters them along the lattice traversals. PISA adds the signatures of discovered intensive clusters to a *signatures table*, which stores the fields and values for each signature. Additional information for each signature is the number of samples in which it was seen, the average number of flows and packets/bytes carrying the signature. A Time To Live (TTL) field associated with the signatures flushes out signatures not seen over many samples. A signature can be logged before it is flushed out.

Consider the four parameters in PISA: N, T, k and the fields of interest in the packets. N is the number of packets sampled in the scan phase. Larger values of N will typically increase the size of the flow lattice and may yield more signatures. However, reasonably sized values for N should be sufficient for finding intensive and persistent signatures. This claim is supported by experiments presented in Section 4. T (threshold for the weight of an intensive cluster) can be specified as a percentage of the traffic. Low values of T may yield many signatures and high values may yield fewer signatures. k is the minimum dimension of a signature. If it is set to low values such as 1 or 2, the number of detected signatures may be very high and many of them may be meaningless. If k is set to too high a value, few or no signatures may be detected. The fields of interest in a packet may include protocol header fields, application level fields and fields that may be defined on the packet content.

Two approaches to determine the number of lattice traversals *I* are: (1) Static: *I*=*L***N*, where *L* is a constant called the lattice factor (2) Dynamic: Use a constant called the exit factor, which is the maximum number of consecutive lattice traversals where no new intensive clusters (or, a few) are seen. Since signatures that are carried by a large number of flows should have a large number of representative sets, they should be found quickly and it may be necessary to traverse only a small percentage of the paths in the lattice to detect them.

3.1 Analysis of PISA

Assume that the number of packets in a scan is *N*, the number of flows in a scan is *f*, and the number of fields of interest is *h*. In each lattice traversal, representative sets are collected and each representative set is grown against the flow table. Any path in the lattice is of length *f*. At every node in the path taken by a lattice traversal, at most *h* comparisons are made between fields of the cluster at the previous node in the path and the newly added flow. The maximum number of representative sets along a path is (h-k+1). The time taken to grow a representative set is proportional to *f*. If there are *I* lattice traversals, the upper bound on the time taken for PISA is O$(I * (f * h + (h-k+1) * f * h))$. Since *h*, *k* are constants and *N* is an upper bound on *f*, the time complexity can be expressed as O(I*N). If the lattice factor is used to determine *I*, the time complexity is O(N^2). If the exit factor is used, *I* is directly proportional to the number of intensive clusters in the sample and the time complexity of the algorithm is O(Number of intensive clusters * *N*).

The space requirements of the algorithm are dominated by the size of the flow table: O(*N*) and the size of the signatures table: O(Number of Signatures**k*). *k* is a small constant. The number of signatures can be controlled through *T* and the TTL for each signature.

PISA does not report *any* signature that does not belong to a cluster of flows in a sample. Hence, there are no false positives. The probability of finding a signature S in a sample is proportional to the number of flows carrying the signature. This can be estimated, but is not presented here for brevity. The likelihood of missing signatures that are carried by a large number of flows is low.

4 Experimental Setups and Results

An implementation of PISA, referred to as the PISA system, repeatedly executes the scan and signature extraction phases. Extracted signatures are stored in the table of signatures that are logged after a fixed number of executions, referred to as a *scan set*. In our experiments, there are 500 executions in a scan set. All signatures fall into two categories: transient and non-transient. A signature is *transient* if it occurs in fewer than 50 scan phases at the end of each scan set. Non-transient signatures can be divided into two categories - persistent and non-persistent. A signature is *persistent* if it occurs in at least 500 scan phases. A signature is *non-persistent* if it is non-transient and occurs in less than 500 scan phases. The *frequency* of a signature is a measure of its occurrence in a scan set. The results reported in this paper consider non-transient signatures only. The value for the TTL field was 500. Significant signatures are defined to be those that have at least 4 dimensions (dimensionality), have a bandwidth consumption of at least 10% (intensity), and occur for at least 500 scans (persistence).

Two kinds of data were used in our experiments[1]: (1) Traces from CAIDA [14] provided a testbed of DDoS attacks and (2) data from the WIDE backbone [13] in the week of Blaster worm outbreak served as a testbed to identify worm spread signature in a mix of traffic.

4.1 CAIDA Trace Data and Results

CAIDA traces contain a wide variety and number (~8000) of attacks and have been analyzed by independent measurements [11]. CAIDA traces are useful in understanding the depth of information that can be captured by the PISA system and served as a useful test-bed to reveal different properties of PISA. The CAIDA trace used in our experiments consisted primarily of backscatter data from nearly two weeks (denoted by Week-I-II). *Backscatter packets* are defined as unsolicited response packets that include ICMP (host unreachable, port unreachable, time exceeded etc.) or TCP packets (syn-ack and reset flows). Details of the experimental setup to capture traces on the /8 network can be obtained in [11]. Since there is a strong correlation between attack packets and backscatter packets, PISA applied to backscatter packets extracts signatures that can be correlated to the signatures for attack flows. For example, if a signature for backscatter flows contains the value SYN-ACK for the TCP flag, the signature in the attack flows will contain the value SYN for the same TCP flag.

The values of the parameters in our experiments (unless otherwise specified) are as follows: $k = 4$, T=10% of N, $L = 20$, $N = 50$. TCP/ICMP/UDP and IP header fields are considered for inclusion in packet signatures. The criterion for transient signatures (50 scans) in the PISA system approximately corresponds to the criteria (attacks lasting for less than a minute) for signatures ignored in [11]. Unless stated otherwise, only persistent signatures were considered for analysis. The purpose of the experiments was to study (a) the effect of different parameters on PISA. (b) the total number, duration and type of attacks observed over Week-I-II. In (a), the use of several values for each parameter in experiments necessitated working with a smaller data sample than Week-I-II. A 21.6 hour long data sample was chosen without bias. This is a sufficiently large data sample to mitigate short-term perturbations in traffic.

To study the effect of varying the number of lattice traversals, I was set to a constant (lattice factor) times the number of flows N. Fig. 2 compares the total number of distinct signatures when lattice factors of 5, 10, 20, and 50 are used for a sample size of 50. Similar results were obtained for sample sizes of 25, 40 and 75 packets. The total number of signatures does not increase significantly for lattice factors of 10, 20 or 50, showing the low rate of false negatives. While the number of signatures is reported here, the actual signatures too did not vary as the lattice factor

[1] The authors acknowledge the Senior Staff at CAIDA for the availability of the traces [14] from their work [11] and MAWI working group [13] for making traces from the WIDE backbone available to us.

was increased. An analysis of the signatures showed that frequency of occurrence of persistent signatures (not shown here) did not change significantly (low rate of false negatives) as the lattice factor varied between 5-50. For example, as the lattice factor is varied between values of 5-50, the frequencies of occurrence of a highly persistent signature were observed to vary between 94-96%. The frequencies of occurrence of another lesser persistent signature were observed to vary between 66-77%, again demonstrating the low rate of false negatives for persistent signatures.

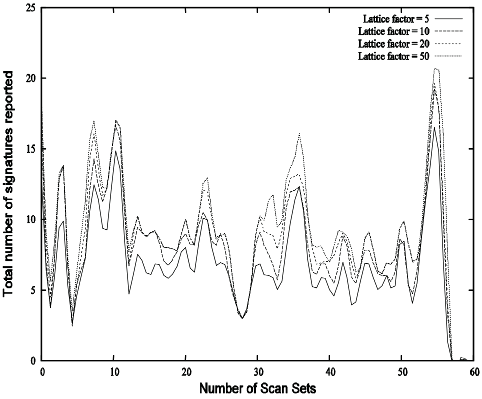


Fig. 2. Number of signatures for various lattice factors

Table 1. Frequency of non-persistent signatures as a percentage of scan set (500 scans) for varying N

Frequency →	10-25%	25-50%	50-75%	>75%
N=10pkts	24	1	1	0
N=25pkts	39	1	1	1
N=50pkts	53	1	1	1
N=75pkts	53	1	1	1

Table 2. Variation in the number of persistent signatures with different values of threshold, T

Threshold Value of T	Number of persistent signatures
5%	33
10%	28
20%	24

Table 3. Distribution of signatures by TCP and ICMP Protocols

Sig. Type	Total	50-120 scans	120-500 scans	500-10K scans	>10K scans19
TCP	1210	283	661	192	19
ICMP	534	250	12	197	30
Total	1744	533	673	489	49

Table 4. Distribution of Signatures by response protocol

Protocol Type	Number of signatures
TCP (total)	321
TCP (RST, RST ACK)	273
TCP SYN ACK	31
TCP Other	17
ICMP (total)	110
Host Unreachable	57
TTL Exceeded	51
Parameter Problem	1
UDP	1

Table1 shows the number of non-persistent signatures after every scan set. The signatures that occurred in more than 25% of the scans did not change significantly as *N* varied. As *N* increases, the number of signatures that occur between 10-25% of the scans increases but is the same for *N*=50, 75. An examination of the signatures showed that the same signatures are extracted as *N* varies between 10-75. Similar results were observed for persistent signatures. The results support our earlier assertion that *N* need not be very large to extract the intensive signatures. In Section 4.2, results using higher values of *N* are presented.

Table 2 shows how *T* can be used to control the number of signatures. As the value of *T* increases from 5% to 20%, the number of signatures decreases since fewer clusters of flows will have a weight greater than *T*.

Having shown the effect of varying values of parameters in PISA, we present results from the execution of PISA on Week-I-II.

Signatures of dimensions between 4-6 were detected in our experiments. About 65% of the signatures are 4-dimensional, about 25% are 5-dimensional and about 10% of the signatures are 6-dimensional. Consider a source sending SYN requests to multiple destinations of which two destinations are prominent. Sample observed signatures are presented below. The values for each field are represented by *a*, *b*, *c*, *d*, *x*, and *y*.

S(5): 5-dimensional, *{(packet_size, a), (src_port, b), (src_addr, c), (type, d), (tcp_flag, SYN)}*

S(4): 4-dimensional, *{(src_port, b), (src_addr, c), (type, d), (dest_addr_1, x)}*

$S_1(6)$: 6-dimensional, *{(packet_size, a), (src_port, b), (src_addr, c), (type, d), (tcp_flag, SYN), (dest_addr_1, x)}*

$S_2(6)$: 6-dimensional, *{(packet_size, a), (src_port, b), (src_addr, c), (type, d), (tcp_flag, SYN), (dest_addr_2, y)}*

$S(4) \subset S_1(6)$ and $S(5) \subset S_1(6)$ and $S(5) \subset S_2(6)$. A comparison of signatures shows: (1) S(4) occurs very infrequently and contains a slightly larger number of flows compared to $S_1(6)$. S(5) occurs more frequently than $S_1(6)$ and $S_2(6)$. (2) The average number of flows / packets in S(5) is more than twice that of either $S_1(6)$ or $S_2(6)$.

For reasons of ease, signatures with the protocol field value of TCP/ICMP will be referred to as TCP/ICMP signatures, respectively. Table 3 shows the frequency of TCP and ICMP signatures. Signatures occurring between 50-500 scans (non-persistent) correspond to attack durations of about 2-20 minutes. Signatures that occur between 500-10,000 scans correspond to attacks that last between 20 minutes-several hours. Signatures occurring in more than 10,000 scans correspond to attacks lasting over several hours and up to several days. The results are consistent with [11], which states that 50% of the attacks are less than 10 minutes in duration, 80% are less than 30 minutes, 90% last less than an hour and the rest span from several hours to a few days.

Distribution of signatures by response protocol is presented in Table 4. ICMP signatures with the error code “TTL Exceeded” occur in about 50% of all ICMP signatures. About 80% of TCP signatures contain the RST/RST ACK flag in the TCP flag field. The total number of TCP signatures is about three times the number of ICMP signatures. This agrees with attacks classified on the response protocol in [11].

Fig. 3 shows plots for the total number of persistent signatures and the number of persistent TCP signatures. ICMP signatures make up the difference between the two plots.

It is difficult to directly compare the number of signatures from the PISA system to the ones reported in [11] because the approach in [11] models the attacks as emanating from groups of flows where each group contains flows all destined to the same IP address. Thus, similar flows to different destination IP addresses are not classified into the same attack. An examination of the signatures from our experiments shows that 85% of the signatures collected by the PISA system did not have destination IP address as a field in the signature because PISA clustered flows to different destinations based on other fields in the packets. This explains why the signatures identified by PISA is about one-fourth of the number seen in [11]. This is significant because PISA enables multiple simultaneous attacks that share similar characteristics (in fields) to be represented by a single signature.

Results may be summarized as follows: (1) 100 million packets in Week-I-II yielded about 1744 signatures. (2) The threshold (*T*) controls the number of intensive signatures. (3) A variety of fields appear in the signatures including protocol, packet size, source address, source port, ICMP error types, TCP flags, destination address and destination port. (4) Intensive signatures can be extracted from small samples of data, thus enabling signatures to be detected quickly. (5) A large number of lattice traversals are not required to detect signatures. (6) Signatures with dimensions varying from 4-6 were extracted from the data. This shows that varying levels of information may be gathered in attacks. (7) The duration and the type of signatures are comparable to those reported in [11].

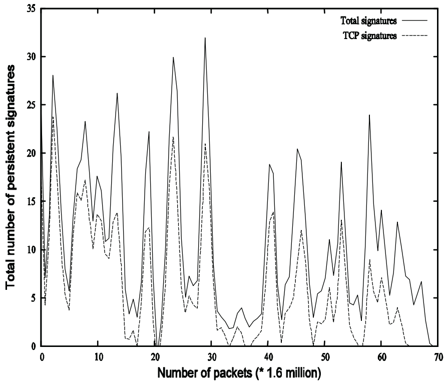


Fig. 3. Total number of persistent signatures and number of persistent TCP signatures every 1.6 million packets

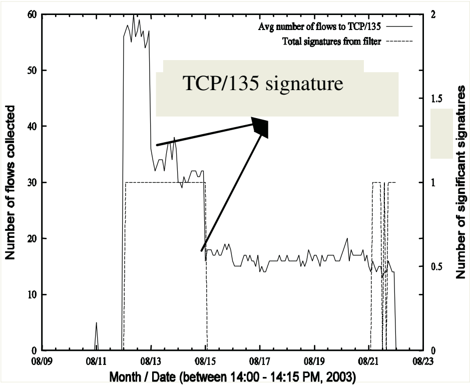


Fig. 4. Persistence (secondary y-axis) & average number of flows (primary y-axis) of the signature of Blaster connection request to TCP/135

4.2 Experience with the Blaster Worm

This section discusses the effect of executing PISA on traffic gathered in the week of the Blaster [1] worm outbreak. Details of the Blaster worm spread and attack phase can be found in [1]. While the worm was active on multiple ports, TCP/port 135 is considered for brevity. Traffic collected during a 15-minute intervals for 2 weeks (9-21 Aug 2003) from the WIDE [13] backbone are analyzed. The attack traffic contributed less than 5% of the bytes over the measured interval.

The following values were used for the PISA parameters: T=9%, k=4. About 25% of the traffic was sampled, which corresponded to 120-250 packets per scan.

Fig. 4 shows the number of significant signatures per scan set and the average number of flows carrying the signature corresponding to attempts to connect to TCP port 135. The connection attempt, that shows up as a persistent and intensive signature was identified as *{(packet_size, 48), (dest_port, 135), (type, TCP), (tcp_flag, SYN)}*. Non-persistent signatures that are supersets of this persistent signature and of the form *{(packet_size, 48), (dest_port 135), (type, TCP), (tcp_flag, SYN), (src_addr, [IP1, IP2 … IPn])}*, where IP1, IP2 … IPn denote different IP addresses, were observed frequently. The figure also shows persistent signatures on 21 and 22 August for RSTP data transfers and ICMP/92 (echo requests with a packet size of 92 bytes). ICMP/92 corresponds to the ping sweep of the Welchia.A [1] worm. Other non-persistent signatures observed corresponded to NNTP and FTP data transfer, and connection requests to web servers. When *T* was reduced to 7%, additional persistent signatures corresponding to RSTP, connect requests to web servers and ICMP/92 (from 19 August onwards) were observed.

5 Conclusions

Signatures represent aggregated characteristics of network flows that is the hallmark of many DDoS attacks and worm spreads. This paper defines characteristic properties of signatures such as dimensionality, intensity, persistence, and distribution, which help define significant signatures that may be of interest while analyzing network traffic. The paper proposes PISA, an efficient algorithm to extract significant signatures from samples of traffic. The fields in the signatures enable filters to be designed to isolate flows suspected of belonging to attacks and the persistence and intensity of signatures can form the basis for rate-limiting the isolated flows. The paper presents experimental results for real network traces that show that PISA works very effectively in extracting signatures for a wide range of attack scenarios. PISA abstracts about 100 million packets participating in denial of service attacks to 1,744 significant signatures. In addition, the results show that PISA effectively extracts the signature for the connection requests corresponding to the Blaster worm in a mix of traffic from a backbone link.

References

[1] CERT, Vulnerabilities, Incidents and Fixes, http://www.cert.org/nav/index_red.html

[2] R. O. Duda and P. E. Hard. Pattern Classification and Scene Analysis. Wiley-Interscience, NY, 1973.

[3] C. Estan, S. Savage and G. Varghese, "Automatically Inferring Patterns of Resource Consumption in Network Traffic", *Proceedings of the ACM SIGCOMM Conference*, Karlsruhe, Germany, August 2003.

[4] H. V. Jagadish, J. Madar, and R.T. Ng, "Semantic Compression and Pattern Extraction with Fascicles", *Proceedings of 25th VLDB*, pp. 186-198, 1999.

[5] A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, New Jersey, 1988.

[6] C. Jin, H. Wang, and K. G. Shin. "Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic", *ACM Conference on Computer and Communications Security (CCS)'2003* , October 2003.

[7] D. Dittrich, "Distributed Denial of Service Attacks/Tools", http://staff.washington.edu/dittrich/

[8] H.-A. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection", *Proceedings of the 13th Usenix Security Symposium (Security 2004)*, San Diego, CA, August, 2004.

[9] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network." *Computer Communications Review* 32:3, July 2002, pp. 62-73.

[10] H. Mannila and H. Toivonen. Level_Wise search and borders of theories in knowledge discovery, *Data Mining and Knowledge* Discovery, 1,3, pp 241-258.

[11] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial of Service Activity", *Proceedings of the 2001 USENIX Security Symposium, Washington D.C., August 2001.*

[12] S. Singh, C. Estan, G. Varghese, and S Savage, "Automated Worm Fingerprinting", *Proceedings of the 6th ACM/USENIX OSDI Symposium*, San Francisco, CA, December 2004.

[13] MAWI Working Group, "Packet traces from WIDE backbone ", http://tracer.csl.sony.co.jp/mawi/

[14] UCSD Network Telescope Backscatter Datasets for February 2001, CAIDA, http://www.caida.org/analysis/security/telescope/

[15] P. Chhabra et. al. XCHOKe: Malicious Source Control for Congestion Avoidance at Internet Gateways. Proceedings of 10th IEEE ICNP, 2002.

FPL-3: Towards Language Support for Distributed Packet Processing

Mihai Lucian Cristea[1], Willem de Bruijn[2], and Herbert Bos[3]

[1] Leiden University, Niels Bohrweg 1, 2333CA
cristea@liacs.nl
[2] Vrije Universiteit Amsterdam, De Boelelaan 1081HV
wdb@few.vu.nl
[3] Vrije Universiteit Amsterdam, De Boelelaan 1081HV,
The Netherlands
herbertb@cs.vu.nl

Abstract. The FPL-3 packet filtering language incorporates explicit support for distributed processing into the language. FPL-3 supports not only generic header-based filtering, but also more demanding tasks, such as payload scanning, packet replication and traffic splitting. By distributing FPL-3 based tasks across a possibly heterogeneous network of processing nodes, the NET-FFPF network monitoring architecture facilitates very high speed packet processing. Results show that NET-FFPF can perform complex processing at gigabit speeds. The proposed framework can be used to execute such diverse tasks as load balancing, traffic monitoring, firewalling and intrusion detection directly at the critical high-bandwidth links (e.g., in enterprise gateways).

Keywords: High-speed packet processing, traffic splitting, network monitoring.

1 Introduction

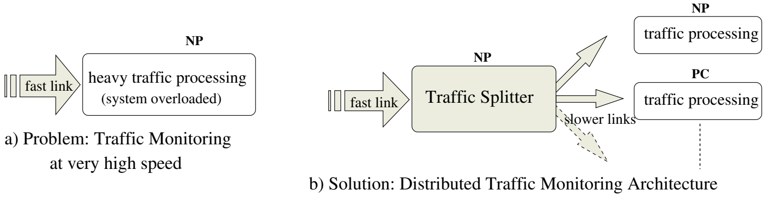
There exists a widening gap between advances in network speeds and those in bus, memory and processor speeds. This makes it ever more difficult to process packets at line rate. At the same time, we see that demand for packet processing tasks such as network monitoring, intrusion detection and firewalling is growing. Commodity hardware is not able to process packet data at backbone speeds, a situation that is likely to get worse rather than better in the future. Therefore, more efficient and scalable packet processing solutions are needed.

It has been recognised that parallelism can be exploited to deal with processing at high speeds. A network processor (NP), for example, is a device specifically designed for packet processing at high speeds by sharing the workload between a number of independent RISC processors. However, for very demanding applications (e.g., payload scanning for worm signatures) more power is needed than any one processor can offer. For reasons of cost-efficiency it is infeasible to develop NPs that can cope with backbone link rates for such applications. An attractive alternative is to increase scalability by exploiting parallelism at a coarser granularity.

R. Boutaba et al.(Eds.): NETWORKING 2005, LNCS 3462, pp. 743–755, 2005.

We have previously introduced an efficient monitoring framework, Fairly Fast Packet Filters (FFPF) [1], that can reach high speeds by pushing as much of the work as possible to the lowest levels of the processing stack. The NIC-FIX architecture [2] showed how this monitoring framework could be extended all the way down to the network card. To support such an extensible programmable environment, we introduced the special purpose FPL-2 language.

In this paper, we exploit packet processing parallelism at the level of individual processing units (NPs or commodity PCs) to build a heterogeneous distributed monitoring architecture: NET-FFPF. Incoming traffic is divided into multiple streams, each of which is forwarded to a different processing node (Fig. 1). Simple processing occurs at the lower levels (root nodes), while more complex tasks take place in the higher levels where more cycles are available per packet. The main contribution of this paper consists of a novel language that explicitly facilitates distribution of complex packet processing tasks: FPL-3. Also, with NET-FFPF we extend the NIC-FIX architecture upwards, with packet transmission support, to create a distributed filtering platform. Experiments show NET-FFPF to be able to handle complex tasks at gigabit line-rate.



a) Problem: Traffic Monitoring at very high speed

b) Solution: Distributed Traffic Monitoring Architecture

Fig. 1. Moving to distributed traffic monitoring

This paper builds on the idea of traffic splitting that was advocated by Markatos *et al.* in [3] and Kruegel *et al.* in [4] for intrusion detection. However, we use it to provide a generic high-speed packet processing environment. Markatos *et al.* focus on packet *header* processing and automatically generate the appropriate code for the splitter (implemented on a network processor) from high-level snort rules. They show that traffic splitting improves the packet processing performance even if the splitter and all processing nodes reside on the same host. The traffic slicer in [4] employs a two-stage approach for intrusion detection where rules for traffic splitting are formed by modelling the attacks. The NET-FFPF implementation resembles the slicer in that it also mangles Ethernet frames to split the traffic. At a more fundamental level, however, NET-FFPF differs from both of the above approaches in that it allows for processing hierarchies that are arbitrarily deep and heterogeneous, whereby each level performs a part of the total computation. Moreover, NET-FFPF offers explicit support for such processing at the language level. By applying the splitter concept in a distributed fashion NET-FFPF can facilitate such diverse tasks as load balancing, traffic monitoring, firewalling and intrusion detection in a scalable manner, e.g., in enterprise gateways.

The remainder of this paper is organised as follows. In Section 2, the architecture of the distributed monitoring system and its supporting language are presented. Section 3 is devoted to the implementation details. The proposed architecture is evaluated in Section 4. Related work is discussed throughout the text and summarised in Section 5. Finally, conclusions are drawn and options for future research are presented in Section 6.

2 Architecture

2.1 High-Level Overview

At present, high speed network packet processing solutions need to be based on special purpose hardware such as dedicated ASIC boards or network processors (see Fig. 1a). Although faster than commodity hardware, solutions based even on these platforms are not sustainable in the long run because of a widening gap between growthrates in networking (link speed, usage patterns) and computing (cpu, main memory and bus speed).

To counter this scalability trend we propose the solution shown in Fig. 1b, which consists of splitting the incoming traffic into multiple sub-streams, and then processing these individually. Processing nodes are organised in a tree-like structure, as shown in Figure 2. By distributing these nodes over a number of possibly simple hardware devices, a flexible, scalable and cost-effective network monitoring platform can be built.

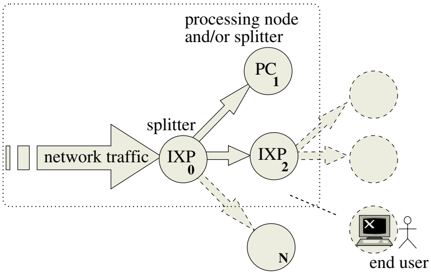


Fig. 2. Distributed system overview

Each node in the system performs a limited amount of packet processing (e.g., filtering, sampling) and may split its incoming stream according to some arbitrary criteria into multiple output streams that are sent to different nodes at higher levels. For example, all TCP traffic is sent to node N_1, all UDP traffic to node N_2. As the traffic arrives at node N_0 at the full link rate, there will be no time for complex packet processing on this node, due to the limited cycle budget. Therefore, at this node we perform only a very simple classification of packets into substreams. Each substream's packets are forwarded to a dedicated node at the next level in the tree. In general, we do not restrict classification to the number of processing nodes in the next level. In other words, it may happen that packets of class X and packets of class Y are sent to the same node at the next level. It also may be necessary to multicast packets to a number of nodes, e.g., a

TCP output stream sent to node N_1 overlaps both an HTTP output stream sent to node N_2 and an SMTP stream sent to node N_3.

The demultiplexing process continues at the next levels. However, the higher we get in the hierarchy, the fewer packets we need to process. Therefore, more complex tasks may be executed here. For instance, we may want to perform signature matching or packet mangling and checksum recalculation. In principle, all non-leave nodes function as splitters in their own rights, distributing their incoming traffic over a number of next level nodes. Optionally, an end-user can check out processing results from the node he/she is interested in using special-purpose applications or third-party tools (e.g., `tcpdump` or `Snort`).

2.2 Distributed Abstract Processing Tree

The introduced networked processing system can be expressed in a distributed abstract processing tree (D-APT) as depicted in Figure 3. The name is derived from a close resemblance to ASTs (abstract syntax trees), as we will see later in this paper. For an easier understanding of the D-APT functionality, we use the following notations throughout the text. A D-APT is a tree composed of individual APTS, each of which has its own dedicated hardware device. An APT is built up of multiple processing elements (e.g., filters) and may be interconnected to other APTS through so-called in-nodes and out-nodes. For example, $N_{0.3}$, $N_{0.5}$ are out-nodes, while $N_{1.1}$, $N_{2.1}$ are in-nodes.

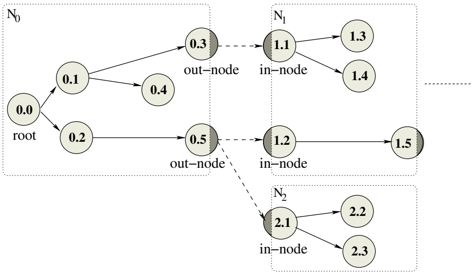


Fig. 3. Distributed Abstract Processing Tree

By ordering the processing nodes, APTS also describe the traffic streams that flow between them. The incoming stream is decomposed into multiple substreams. Simple processing is performed at the lower levels, on the left, while more complex processing happens in the higher levels, on the right (see Fig. 3). Therefore, the amount of traffic and per packet processing ability are well balanced on each tree node.

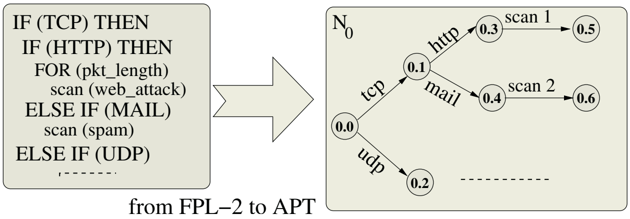
As an APT represents a traffic splitting as well as a processing AST, a stringent requirement is that processing at any level N continues exactly where it left off at level $N - 1$. We can achieve this by explicitly identifying the breakpoint.

We note that the performance of the whole distributed monitoring system is determined by the total number of the processing nodes, the processing power of each node, as well as the distribution of tasks over the nodes.

2.3 The FPL–3 language

As our architectural design relies on explicit breakpointing, we needed to introduce this functionality into our framework. With FPL-3 we adopted a language-based approach, following our earlier experiences in this field. We designed FPL-3 specifically with these observations in mind: first, there is a need for executing tasks (e.g., payload scanning) that existing packet languages like BPF [5], Snort [6] or Windmill [7] cannot perform. Second, special purpose devices such as network processors can be quite complex and thus are not easy to program directly. Third, we should facilitate on-demand extensions, for instance through hardware assisted functions. Finally, security issues such as user authorisation and resource constraints should be handled effectively. The previous version of the FPL-3 language, FPL-2, addressed many of these concerns. However, it lacked features fundamental to distributed processing like packet mangling and retransmission.

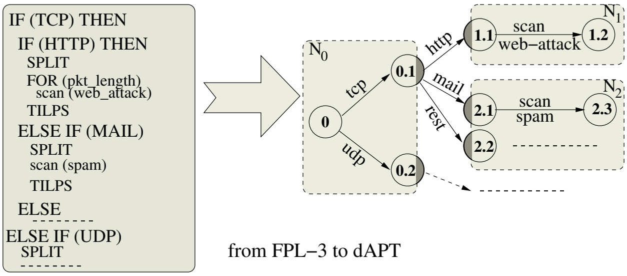
We will introduce the language design with an example. First, a program written for a single machine (N_0) is introduced in Figure 4. Then, the same example is 'mapped' onto a distributed abstract processing tree by using FPL-3 language extensions in Figure 5.



```
IF (TCP) THEN
  IF (HTTP) THEN
    FOR (pkt_length)
      scan (web_attack)
  ELSE IF (MAIL)
     scan (spam)
 ELSE IF (UDP)
    -------
```

Fig. 4. Traffic monitoring program mapped onto APT

Fig. 4 shows how the full stream is split at the root node $N_{0.0}$ into two big substreams: `TCP` and `UDP`. Then, the `TCP` sub-stream is again split into two substreams, `http` and `mail`, by the intermediate node $N_{0.1}$. Because each of these require heavy computation, they are forwarded to the leaves: $N_{0.5}$ and $N_{0.6}$, respectively.



```
IF (TCP) THEN
  IF (HTTP) THEN
    SPLIT
    FOR (pkt_length)
      scan (web_attack)
    TILPS
  ELSE IF (MAIL)
    SPLIT
    scan (spam)
    TILPS
  ELSE
    --------
ELSE IF (UDP)
  SPLIT
  --------
```

Fig. 5. mapping an APT to a D-APT using FPL-3's SPLIT command

When a task requires a large amount of *per-packet* processing power (e.g., a full packet scan for a worm), it becomes infeasible to perform this task on a single machine when network speeds go up. However, the layered style that all protocols reside onto a basic Ethernet frame gives enough parallelism for an eventually distributed processing environment. Thus, we give the same example, written using the SPLIT extension for a distributed processing environment and we take the hardware depicted in Figure 2 as environment. For the sake of simplicity we limit our tree to two levels. The program is mapped into the D-APT shown in Figure 5 by taking into account both the user request (our program) and a hardware description. As Figure 6 shows, the FPL-3 compiler translates the program into multiple output objects, one for each hardware device. Currently, the object files need to be transferred and loaded by hand, but we plan to incorporate a management system into NET-FFPF that will take care of loading and releasing the runtime code as well as fetching the results of each node as one.

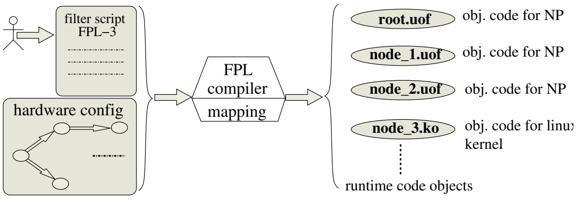


Fig. 6. User compiles an FPL-3 filter expression

3 Implementation Details

NET-FFPF builds on FFPF, a monitoring framework designed for efficient packet processing on commodity hardware, such as PCs. FFPF offers support for commonly used packet filtering tools (e.g., tcpdump, snort, libpcap) and languages (like BPF), as well as for special purpose hardware like network processors (NPs). However, it is a single-device solution. NET-FFPF extends it with distributed processing capabilities through language constructs. Currently, we have support for two target platforms: ① IXP1200 network processors and ② off-the-shelf PCs running Linux. We briefly introduce the first in the next section.

3.1 Network Processors

Network processors are designed to take advantage of the parallelism inherent in packet processing tasks by providing a number of independent stream processors (μEngines) on a single die. The Radisys ENP 2506 network card that was used to implement NET-FFPF is displayed in Figure 7. For input, the board is equipped with two 1Gb/s Ethernet ports ①. The card also contains a 232 MHz Intel IXP1200 network processor with 8 MB of SRAM and 256 MB of SDRAM ② and is plugged into a 1.2 GHz PIII over a PCI bus ③. The IXP is built up of a single StrongARM processor running Linux and six μEngines running no operating system whatsoever.

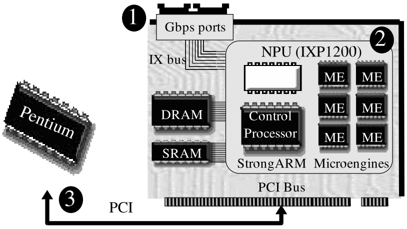


Fig. 7. Main components of the ENP-2506 board

3.2 The FPL-3 language

The FFPF programming language (FPL) was devised to give the FFPF platform a more expressive packet processing language than available in existing solutions. The latest version, FPL-2, conceptually uses a register-based virtual machine, but compiles to fully optimised object code. However, FPL-2 was designed for single node processing. We now introduce its direct descendant, FPL-3, which extends FPL-2 with constructs for distributed processing.

operator-type	operator
Arithmetic	`+, -, /, *, %, --, ++`

Assignment	`=, *=, /=, %=, +=, -=`
	`<<=, >>=, &=, ^=, |=`

Logical / Relational	`==, !=, >, <, >=, <=,`
	`&&, ||, !`

Bitwise	`&, |, ^, <<, >>`

statement-type	operator
if/then/else	IF (expr) THEN stmt1 ELSE stmt2 FI
for()	FOR (initialise; test; update) stmts; BREAK; stmts; ROF
external function	INT EXTERN(filter, input, output)
hash()	INT HASH(start_byte, len, mask)
return a value	RETURN (val)
transmit to	TX(table_type, table_index) or TX(table_type, id, table_index)
split the code	SPLIT; stmts; TILPS or SPLIT(node_index); stmts; TILPS

Data type	syntax
Register n	R[n]
Memory location n	M[n]
Packets access:	
-byte $f(n)$	PKT.B[$f(n)$]
-word $f(n)$	PKT.W[$f(n)$]
-double word $f(n)$	PKT.DW[$f(n)$]
-bit m in byte n	PKT.B[n].U1[m]
-nibble m in byte n	PKT.B[n].U4[m]
-bit m in word n	PKT.W[n].U1[m]
-byte m in word n	PKT.W[n].U8[m]
-bit m in dword n	PKT.DW[n].U1[m]
-byte m in dword n	PKT.DW[n].U8[m]
-word m in dword n	PKT.DW[n].U16[m]
-macro	PKT.macro_name
-ip proto	PKT.IP_PROTO
-ip length	PKT.IP_LEN
-etc.	customised macros

Fig. 8. FPL-3 language constructs

The FPL-3 syntax is summarised in Figure 8. It supports all common integer types and allows expressions to access any field in the packet header or payload in a friendly manner. An extensible set of macros implements a shorthand form for well-known fields, so that instead of asking for bytes nine and ten to obtain the IP header's protocol field, a user may abbreviate to '`IP_PROTO`', for instance. Moreover, offsets in packets can be variable, i.e. determined by an expression. Execution safety is guaranteed by virtue of both compile-time and run-time boundary checks. As illustrated in Figure 8, most of the operators are designed in a way that resembles well-known imperative languages such as C or Pascal. We will briefly explain those constructs that are not intuitively clear.

EXTERN() construct. External functions allow users to call efficient C or hardware assisted implementations of computationally expensive functions, such as checksum calculation, or pattern matching. The concept of an 'external function' is another key to speed and system extensibility.

HASH() construct. Applies a hash function to a sequence of bytes in the packet data. This function is hardware-accelerated on IXP1200 network processors.

TX() construct. The purpose of this construct is to schedule the current packet for transmission. Currently, this operation involves overwriting the Ethernet destination address (ETH_DEST) of a packet with an entry from the MAC_table (TX_MAC). A simple use of TX() is illustrated in the example below:

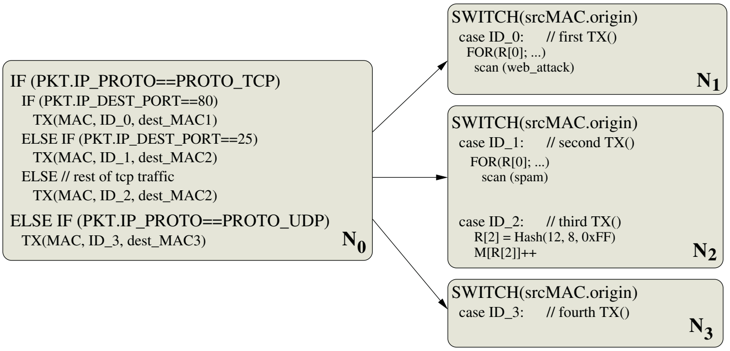
```
TX_MAC[3] = {00:00:E2:8D:6C:F9, 00:02:03:04:05:03, 00:02:B3:50:1D:7A};
 // extracted by the compiler from the configuration file
IF (PKT.IP_PROTO == PROTO_TCP) // if pkt is TCP
  THEN TX (Mac, 2);            // schedule it to be forwarded to the 3rd
  ELSE TX (Mac, 1);            // or 2nd MAC address from the TX_MAC table
FI
```

The example shows the first TX parameter to select a table (MAC or another field, such as IP_DEST, in a future implementation) and the second parameter to be the index into the table. Note that by inserting multiple TX() calls into the same program we can easily implement packet replication and load-balancing.

SPLIT() construct. To explain SPLIT we will step through the example in Figure 5. When trying to match the given FPL-3 filter to a distributed system, the compiler detects the SPLIT construct. SPLIT tells the compiler that the code following and bounded by its TILPS construct can be split off from the main program. The example script is split into subscripts as follows: one script for the splitter node N_0, and three more for each processing node N_1, N_2 and N_3, as shown in Fig. 9. Then, the scripts are compiled into object code by the native compilers.

The current implementation is based on Ethernet header mangling (driven by TX constructs). A packet's Ethernet destination address (ETH_DEST_ADDR) is overwritten to contain the next hop in the network. Recall that one of the NET-FFPF requirements is that processing at level N continues exactly where it had broken off at level $N-1$. The way we implemented this in the Ethernet implementation is by using the Ethernet source address to identify the out-node at level $N-1$. However, there is no need to use all six bytes of the source address for this purpose. For this reason we split the source address (ETH_SRC_ADDR) into two identifiers: *processing state* (four bytes) and *origin indicator* (two bytes).

The origin indicator specifies the out-node of the previous level (allowing for 64K out-points), while the 32 bit state field can be used in an application-specific way. It allows nodes at level $N-1$ to pass a limited amount of information to the next processing node. It is beyond the scope of this paper to discuss the use of the state field in



```
IF (PKT.IP_PROTO==PROTO_TCP)
  IF (PKT.IP_DEST_PORT==80)
    TX(MAC, ID_0, dest_MAC1)
  ELSE IF (PKT.IP_DEST_PORT==25)
    TX(MAC, ID_1, dest_MAC2)
  ELSE // rest of tcp traffic
    TX(MAC, ID_2, dest_MAC2)
ELSE IF (PKT.IP_PROTO==PROTO_UDP)
  TX(MAC, ID_3, dest_MAC3)
```

```
SWITCH(srcMAC.origin)
 case ID_0:     // first TX()
   FOR(R[0]; ...)
     scan (web_attack)
```

```
SWITCH(srcMAC.origin)
 case ID_1:     // second TX()
   FOR(R[0]; ...)
     scan (spam)

 case ID_2:     // third TX()
   R[2] = Hash(12, 8, 0xFF)
   M[R[2]]++
```

```
SWITCH(srcMAC.origin)
 case ID_3:     // fourth TX()
```

Fig. 9. SPLIT in detail

detail. As shown in Figure 9, we can now efficiently continue the computation at level N, by using a `switch` statement on the origin indicator to jump to the appropriate point to resume. Observe that a `switch` statement is only needed if more than one out-node is connected to this in-node. Also observe that although we have not implemented this, it may be possible to generalise NET-FFPF beyond subnets by basing the splitter functionality on a tunnelled approach or by overwriting IP header fields, instead.

The compiled example program is executed as follows. The code at the root node (N_0) deals only with selective forwarding. Any packet that matches one of the IF statements has its Ethernet address fields modified and is then forwarded to the next-level nodes. The other nodes will only have to process these forwarded packets. Node N_2 for instance, receives two classes of packets forwarded from node N_0. As illustrated in Figure 9, the classes are identified by the `origin` indicator embedded in the `ETH_SRC_ADDR` field.

Note that by passing the optional argument 'node_index' to `SPLIT` a user can force packets to be forwarded to a specific node, as in the example shown in Section 4. This can be useful when a node has special abilities well-suited to a given task, e.g., hardware-accelerated hashing.

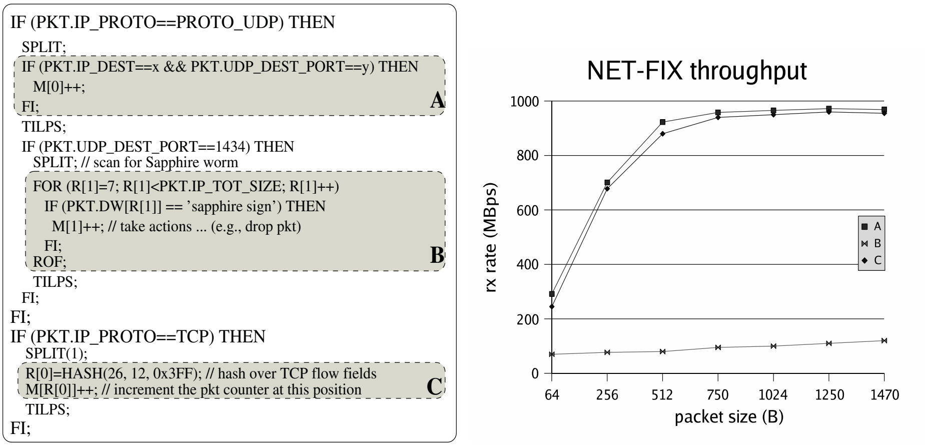
Memory Data Addressing. The FPL-3 language hides most of the complexities of the underlying hardware. For instance, users need not worry about reading data into a μEngine's registers before accessing it. Similarly, accessing bytes in memory that is not byte addressable is handled automatically by the compiler. When deemed useful, however, users may *choose* to expose some of the complexity: it is, for instance, wise to declare additional arrays in fast hardware when that is available, like in the IXP1200's on-board SRAM. Note that NET-FFPF has no inter-node shared memory. Synchronising data across nodes is too costly relative to the high-speed packet processing task itself. To guarantee safe execution enough memory is allocated at each node to support the whole program, not just the subtask at hand. In the Ethernet implementation, limited data sharing is made possible by overwriting a now unused block in the `ETH_SRC_ADDR` field to communicate a 32 bit word *processing state* to the next hop.

3.3 The FPL-3 compiler

The FPL-3 source-to-source compiler generates straight C target code that can be further handled by any C compiler. Programs can therefore benefit from the advanced optimisers in the Intel μEngine C compiler for IXP devices and `gcc` for commodity PCs. As a result, the object code will be heavily optimised even though we did not write an optimiser ourselves. In the near future FPL-3-compiler support will be extended to the latest Intel NP generation (e.g., IXP2400, IXP28xx).

4 Evaluation

To evaluate NET-FFPF, we execute the filter shown in Figure 10.a on various packet sizes and measure throughput (Fig. 10.b). This filter is mapped onto a distributed system as shown in Figure 2 and the compilation results are the code objects of the three subfilters highlighted in same picture.



```
IF (PKT.IP_PROTO==PROTO_UDP) THEN
  SPLIT;
  IF (PKT.IP_DEST==x && PKT.UDP_DEST_PORT==y) THEN
    M[0]++;                                                  A
  FI;
  TILPS;
  IF (PKT.UDP_DEST_PORT==1434) THEN
    SPLIT; // scan for Sapphire worm
    FOR (R[1]=7; R[1]<PKT.IP_TOT_SIZE; R[1]++)
      IF (PKT.DW[R[1]] == 'sapphire sign') THEN
        M[1]++; // take actions ... (e.g., drop pkt)
      FI;
    ROF;                                                     B
    TILPS;
  FI;
FI;
IF (PKT.IP_PROTO==TCP) THEN
  SPLIT(1);
  R[0]=HASH(26, 12, 0x3FF); // hash over TCP flow fields
  M[R[0]]++; // increment the pkt counter at this position   C
  TILPS;
FI;
```

ⓐ **Filter** ⓑ **Benchmark**

Fig. 10. Filtering results

We note that only A is a traditional per-packet counter. The other two gather per-flow information. As the hash function used in C utilises dedicated hardware support, we push (by providing the parameter `node_index`) the C filter onto the same hardware node as A filter – N_1. In B, a loop iterates over the whole UDP packet payload. While the filters A and C are easily performed on an NP even at high speed (1Gb/s), the B filter incurs so much processing that even an IXP1200 network processor cannot handle its stream at such speed. As the processing results show, using this specific hardware, we can process the full packet data up to only ca. 100Mb/s. Therefore, we let the filter split to the second node – N_2 and we can successfully process all the packets for a particular UDP port (assuming the packets related to a specific worm are within

100Mb/s bandwidth). If more bandwith needed, then more processing nodes have to be involved.

As illustrated in Fig. 10, just as for the B filter, throughput also drops for the simple filters A and C when processing smaller packets. However, these drops occur for a different reason, namely because the *receiving* μEngine simply cannot keep up.

Demonstrating the simplicity a concise special purpose language like FPL-3 brings, a naive load balancing program for a web-server is shown below. A hash over the flow fields of each incoming packet determines to which node the packet is forwarded. In a handful lines of code the web traffic is split into three equal substreams.

```
IF (PKT.IP_PROTO==TCP && PKT.IP_DEST_PORT==80) THEN
R[0] = hash(flowfields)%3;
SPLIT(R[0]);                    // thus, the main stream is equally distributed
 <scan for web-traffic worms>   // across of 3 processing nodes
```

5 Related work

Using traffic splitters for increased packet processing efficiency on a single host was previously explored in [3] and [4]. Our architecture differs in that it supports distributed and heterogeneous multi-level processing. In addition, we provide explicit language support for this purpose. As shown in [8], it is efficient to use a source-to-source compiler from a generic language (Snort Intrusion Detection System) to a back-end language supported by the targeted hardware compiler (Intel μEngineC). We propose a more flexible and easy to use language as front-end for users. Moreover, our FPL-3 language is designed and implemented for heterogeneous targets in a distributed multi-level system.

Many tools for monitoring are based on BPF in the kernel [5]. Filtering and processing in network cards is also promoted by some Juniper routers [9]. However, they lack features introduced in NET-FFPF such as extended language constructs, in-place packet handling and a distributed processing environment. BPF+ [10] shows how an intermediate representation of BPF can be optimised, and how just-in-time-compilation can be used to produce efficient native filtering code. FPL-3 relies on gcc's optimisation techniques and on external functions for expensive operations.

Like FPL-3 and DPF, Windmill protocol filters also target high-performance by compiling filters into native code [7]. And like MPF, Windmill explicitly supports multiple applications with overlapping filters. However, compared to FPL-3, Windmill filters are fairly simple conjunctions of header field predicates.

Nprobe is aimed at monitoring multiple protocols [11] and is therefore, like Windmill, geared towards spanning protocol stacks. Also, Nprobe focuses on disk bandwidth limitations and for this reason captures as few bytes of the packets as possible. NET-FFPF has no *a priori* notion of protocol stacks and supports payload processing.

The SCAMPI architecture also pushes processing to the NIC [12]. It assumes that hardware can write packets immediately into host memory (e.g., by using DAG cards [13]) and implements access to packet buffers through a userspace daemon. SCAMPI does not support user-provided external functions, powerful languages such as FPL-3 or complex filtergraphs.

Related to the distributed architecture of NET-FFPF are the Lobster EU project and Netbait Distributed Service [14] that aim at European-scale passive monitoring and at planetary-scale worm detection, respectively. Netbait, for instance, targets high-level processing using commodity hardware. Therefore, these initiatives could benefit from using the FPL-3 language and its NET-FFPF execution environment as low-level layer.

6 Conclusions and Future Work

This paper presented the NET-FFPF distributed network processing environment and its FPL-3 programming language, which enable users to process network traffic at high speeds by distributing tasks over a network of commodity and/or special purpose devices such as PCs and network processors. A task is distributed by constructing a processing tree that executes simple tasks such as splitting traffic near the root of the tree while executing more demanding tasks at the lesser-travelled leaves. Explicit language support in FPL-3 enables us to efficiently map a program to such a tree. The experimental results show that NET-FFPF can outperform traditional packet filters by processing at Gbps linerate even on a small-scale (two node) testbed.

In the future, we plan to extend NET-FFPF with a management environment that can take care of object code loading and program instantiation. A first version of this management subsystem will act only when a user issues a recompile request. We envision a later version to be able to automatically respond to changes in its environment like the increase of specific traffic (e.g., tcp because of a malicious worm) or availability of new hardware in the system (e.g., a system upgrade). We also plan to have the FPL-3 compiler optimise code placement. As a result, tasks that are known to be CPU intensive – such as packet inspection, hashing or CRC generation – will be automatically sent to optimal target machines, for instance those with hardware assisted hash functions (NPs).

Acknowledgements

This work was supported by the EU SCAMPI project IST-2001-32404, and the EU LOBSTER project, while Intel donated the network cards.

References

1. Bos, H., de Bruijn, W., Cristea, M., Nguyen, T., Portokalidis, G.: FFPF: Fairly Fast Packet Filters. In: Proceedings of OSDI'04, San Francisco, CA (2004)
2. Nguyen, T., de Bruijn, W., Cristea, M., Bos, H.: Scalable network monitors for high-speed links: a bottom-up approach. In: Proceedings of IPOM'04, Beijing, China (2004)
3. Charitakis, Anagnostakis, Markatos: An active traffic splitter architecture for intrusion detection. In: Proceedings of 11th IEEE/ACM MASCOTS, Orlando, Florida (2003)
4. Kruegel, C., Valeur, F., Vigna, G., Kemmerer, R.: Stateful intrusion detection for high-speed networks. In: Proceedings of the IEEE Symposium on Security and Privacy. (2002)

5. McCanne, S., Jacobson, V.: The BSD Packet Filter: A new architecture for user-level packet capture. In: Proceedings of the 1993 Winter USENIX conference, San Diego, Ca. (1993)
6. Roesch, M.: Snort: Lightweight intrusion detection for networks. In: Proceedings of the 1999 USENIX LISA Systems Adminstration Conference. (1999)
7. Malan, G.R., Jahanian, F.: An extensible probe architecture for network protocol performance measurement. In: Computer Communication Review, ACM SIGCOMM. (1998)
8. Charitakis, I., Pnevmatikatos, D., Markatos, E.: Code generation for packet header intrusion analysis on the ixp1200 network processor. In: SCOPES 7th International Workshop. (2003)
9. Thomas, T.M.: Juniper Networks Router Architecture. In: Juniper Networks Reference Guide: JUNOS Routing, Configuration, and Architecture. (2003)
10. Begel, A., McCanne, S., Graham, S.L.: BPF+: Exploiting global data-flow optimization in a generalized packet filter architecture. In: In Proceedings of ACM SIGCOMM, Boston (1999)
11. Moore, A., Hall, J., Kreibich, C., Harris, E., Pratt, I.: Architecture of a network monitor. in proc. of PAM'03 (2003)
12. Polychronakis, M., Markatos, E., Anagnostakis, K., Oslebo, A.: Design of an application programming interface for ip network monitoring. In: IEEE/IFIP NOMS, Seoul (2004)
13. Cleary, J., Donnelly, S., Graham, I., McGregor, A., Pearson, M.: Design principles for accurate passive measurement. In: Proceedings of PAM, Hamilton, New Zealand (2000)
14. Chun, B., Lee, J., Weatherspoon, H.: Netbait: a distributed worm detection service (2003)

Efficient Deployment of Honeynets for Statistical and Forensic Analysis of Attacks from the Internet

Stephan Riebach, Erwin P. Rathgeb, and Birger Toedtmann

University Duisburg-Essen,
Computer Networking Technology Group,
Institute for Experimental Mathematics,
Ellernstr. 29, 45326 Essen, Germany
{riebach, erwin.rathgeb, btoedtmann}@exp-math.uni-essen.de

Abstract: The use of honeynets as a means to detect and observe attacks originating from the Internet as well as to allow forensic analysis is a technique that has received increasing attention in the research community. However, it has not yet been investigated how effective honeynets are and to what extent their efficiency can be actively improved. Therefore, after a short introduction to the honeynet concept and its implementation options, a case study will be presented providing some insight into this issue. For this case study, a honeynet has been implemented and a multilevel escalation strategy has been defined and employed to clarify to what extent the detected attacks represent just the "average" level of malware activity and to what extent honeynet owners can actively attract attacks or even influence specific types of attacks.

Keywords: Security, Forensics, Honeypots, Honeynets.

1 Introduction

Along with the increasing penetration of powerful personal computers and the rapid evolution of the internet, the issue of malware and hacker attacks has exploded at the same pace. To cope with this issue, appropriate protection tools, like e.g. firewalls or intrusion detection systems (IDS) are available for all segment sizes, starting from single PCs at home to large corporate networks. However, there is always a tradeoff between security and effort expended, against usage restrictions. In order to be able to balance these conflicting issues for a given scenario, a sound and fairly detailed risk assessment is crucial – both with respect to the probability of specific incidents and with respect to their sophistication. The latter is important, as the threats encountered in today's networks range from blind, fully automated worm and virus activities, via attacks with prefabricated scripts requiring no specific knowledge to highly specific and targeted attacks by expert hackers.

For obvious reasons, it is rather difficult to perform systematic measurements and observations in this area in real life production networks. Therefore, artificial networks specifically set up to observe and document attack activities for offline analysis have been proposed which are now commonly known as "honeynets". Over

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 756 – 767, 2005.

the last few years, this concept has been evolved and refined significantly and is becoming more widely used now. The information gathered in honeynets has already proven quite useful in the forensic analysis of attack activities and has provided valuable insight into various attack mechanisms.

After some intrusions in our own network, and based on the positive results published so far, we have implemented a honeynet in order to evaluate this concept. In addition to actually developing an understanding about the frequency and sophistication of attacks threatening our network, our goal was also to gain some insight into the factors determining the "efficiency" of a honeynet, i.e. its ability to actually attract attacks. This is especially interesting if the honeynet is used not to gather information, but instead as a decoy to divert attackers away from a production network operating in parallel. Even though a white paper [HON04] suggests that it is sufficient to just activate the honeynet as it will be found and attacked without further activities necessary, we wanted to find out to what extent the attractivity of a honeynet can possibly be influenced by the honeynet owner. Therefore we defined a phased escalation strategy which increasingly exposed our honeynet to the internet and observed the results in a field study whose first results will be presented in this paper.

After a short review of the honeynet concept in general and its various implementation options, we will describe our honeynet setup in some detail. We will also define and motivate the various steps of our escalation study before presenting some results with respect to the quantity and quality of the detected attack activities as well as with respect to the escalation study in particular.

2 Honeynets – Goals, Concepts and Implementation

The term "honeynet" was coined by a group of security experts organized in the "Honeynet Project" (www.honeynet.org). This group promotes the development and application of honeynet concepts and is the main source of the definitions used in this section.

The basic idea that led to the development of honeynets was to detect, observe and document the activity of hackers inside a computer network. Honeynets are highly specialized, artificial networks which have to be kept strictly separate from the actual production networks, have no real users – and thus no real traffic activity – and don't contain any real information (user data). To be able to observe attacks, honeynets have to be vulnerable to a certain extent which means that they cannot be strictly protected by firewalls and that their systems should at least show some of the common vulnerabilities. Honeynets are highly controlled which means that elaborate monitoring and logging facilities capture and document all activity to provide comprehensive data for forensic analysis. Because they are artificial, all traffic in a honeynet is by definition suspicious, and traffic originating from a host in a honeynet is an indication that this system has likely been taken over.

In addition to the "honeypot" computers to be scanned, probed or attacked, a "data capture" function is required to make the honeynet useful. In addition to storing all data packets for offline forensic analysis, online monitoring with host and network based Intrusion Detection Systems (IDS) is useful to provide immediate notification about ongoing attacks as well as a basis for targeted forensic analysis. The data capture function can be distributed among several computers (including also the honeypots) or concentrated in a centralized device.

Because honeynets are intentionally vulnerable, so called “data control” mechanisms must be implemented to ensure that intruders cannot misuse compromised honeypots for further attacks. There are several ways to perform data control, e.g., limiting the outgoing bandwidth, restrictive outgoing packet filtering or, adding packet loss and high delays to outgoing connections [HON01]. Because it is particularly important that only the honeypots are visible and accessible for intruders, the data control and capture functions have to be hidden from intruders in order not to reveal the honeynet character of the network. In addition they have to be protected against any manipulation.

The honeynet concept has evolved significantly over the past few years, in particular with respect to implementing data capture and control functions [HON03a]. There is a broad spectrum of realization options for honeynets ranging from software emulating specific aspects of operating systems, applications and services (e.g. Honeyd, see www.honeyd.org) to real networks with hosts providing real services and applications. Whereas simple emulations allow only limited interaction, honeynets with live systems allow full interaction. However, the latter require significantly more effort for setup, configuration and maintenance.

3 Honeynet Setup for the Case Study

In this case study, the honeynet architecture shown in Fig. 1 was used with 5 honeypots connected via a 100baseT hub. The honeynet was connected to the internet via a router (dual homed Linux machine) providing the data control function. We used packet filtering and additional bandwidth limitations for the outgoing traffic to avoid enabling successful attacks to be launched from compromised honeypots.

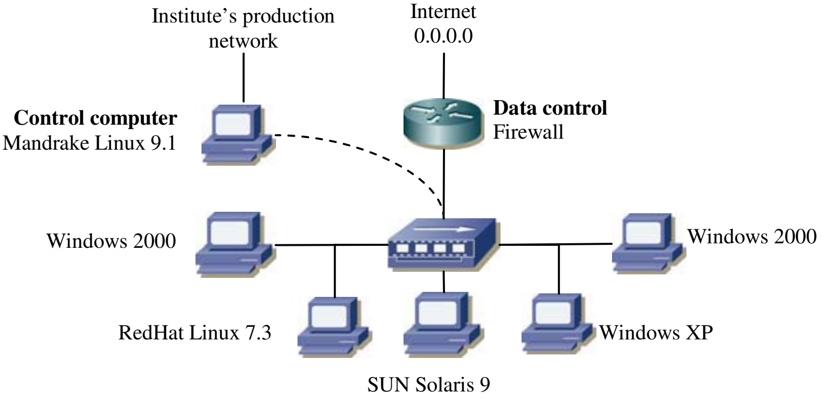


Fig. 1. The honeynet setup for the case study

The control computer was set up with two network interfaces. A modified network cable was used to connect one of these interfaces to the honeynet. By modifying the cable as described in [ICO02], the control computer could receive all traffic from the honeynet but could (physically) not transmit any packets towards the honeynet. Therefore, it was fairly impossible for intruders in the honeynet to detect the presence of this machine and subsequently to attack it. Furthermore, the log data and reports collected on this machine could not be modified from the honeynet, preventing attackers from covering their tracks. Due to these precautions, it was possible to connect the other interface of the control computer to a production network for maintenance and remote data retrieval.

3.1 The Honeypots

As shown in Fig. 1, 2 of the honeypots were configured with Windows 2000 operating system , 1 with Windows XP, 1 with RedHat Linux 7.3 and the last with Solaris 9. This mix of operating systems was chosen because it is fairly typical for our production networks. Microsoft's operating systems are commonly used in the client desktop environment (CDE), with the migration from Windows 2000 to Windows XP starting at the time of our experiments. Unix-based systems (Linux and SUN Solaris) are also used in the production network for personal desktops and typically also as servers providing common services, e.g. http, ftp or nfs.

With respect to the vulnerability of the honeypots we updated to a patch level which was fairly typical for an environment where there is only limited central administration of the systems and the users have to take responsibility for their systems themselves. This means that the systems were not deliberately kept completely unmaintained and vulnerable to make the honeynet character of the network not too obvious. However, not all currently available security patches had been installed to also give also attackers using standard exploits (prefabricated attack scripts) a chance for success. The Windows systems, e.g., got upgraded with the current service pack and the patch for the RPC security leak published in July 2003.

Because of the heterogeneous operating systems, different Host Intrusion Detection Systems (HIDS) had to be used. The freeware tool "Tripwire" [TRIP04] was installed on the Linux system[1], "AIDE" [LETH04] was installed on the Solaris system. Because there was no appropriate freeware HIDS available for Windows the software "InstallWatch" was used there as an alternative. This software is originally intended to monitor installation routines on Windows systems. Therefore, it maintains a database of all system files, the registry and other user selected files and reports the changes after completing an installation. By installing small programs in regular intervals, a so called "poor man Tripwire system" [FLOYD00] is realized. Since the honeypots were not accessible remotely from the production network for security reasons the log files of all HIDSs were collected manually in regular intervals.

Complete images of the software installations of all honeypots were saved for all phases of the escalation study. Therefore, a compromised system could be restored to its original state with minimum effort. Before cleaning up a compromised system, we also saved a complete image for offline analysis and possible reinstallation for further observation.

[1] "Tripwire" is no freeware for Solaris and Windows.

3.2 Traffic Monitoring and Data Capturing

The control computer was responsible for monitoring and capturing all network traffic in the honeynet. For traffic monitoring, the Network Intrusion Detection System (NIDS) "SNORT" [ROE04] was installed to identify known attack signatures in the honeynet traffic continuously and in real time. The SNORT log files were automatically archived once a day. Since they contain all incidents in chronological order only, they were automatically processed locally on the control computer by "SnortSnarf" [SIL03] which produced formatted statistical reports providing, e.g., an overview of the 10 most frequent targets, sources and attack signatures as well as statistics on all detected attack signatures sorted by severity classification and frequency. These statistics presented in HTML were automatically published on a web server on the control computer and could be remotely inspected from the production network. In addition, the major statistics files were automatically sent to the honeynet operator once a day.

For data capturing, the software utility "tcpdump" [TCP04] was deployed. With tcpdump all data traffic occurring in the honeynet was saved into daily dump files including all protocol overhead (addresses, etc.) from OSI layer 2 upwards. This high volume data could be retrieved remotely from the production network for offline analysis and archiving. To assure the permanent availability of these vital honeynet components, SNORT and tcpdump were monitored by using "Daemontools" [BER02] and automatically restarted after irregular shutdowns to avoid data loss.

3.3 Maintenance and Data Analysis

During normal operation, the honeynet generated roughly 1 Mbyte of SNORT log data and 75 Mbyte of tcpdump logs per day. This raw data was completely archived for statistical and forensic analysis. The statistical evaluation was highly automized as described above. The port scan log files generated were transformed into the CSV-format for detailed analysis in MS Excel. The automatic formatting and publishing of the SNORT logs allowed for a quick inspection and gave indications about potentially successful attacks which were then followed up. In addition, the Host Intrusion Detection Systems (HIDS) of the honeypots were collected and inspected on a daily basis so that a compromised honeypot could be identified rather quickly. In addition, sporadic in-depth control and analysis of the honeypots was performed, to minimize the probability of undetected attacks.

Manual forensic analysis was performed in several cases where successful (non-automated) attacks could be detected. For manual inspection of the tcpdump log data, the programs "tcptrace" [OST04] was used to identify successful TCP connections related to successful attacks. "Ethereal" [ETH04] was then used to fully decode the packets of interesting connections up to the application level.

4 The Phased Escalation Strategy

The experience reported by honeynet operators indicates that it is sufficient to just connect a honeynet to the internet and it will be found and attacked almost immediately [HON01]. We wanted to find out if the operator of a honeynet can actively influence the frequency and type of attacks. Therefore, we defined a four step escalation strategy for making our honeynet visible in the internet as follows:

In **phase 1**, the honeypots just had a basic installation of the operating system and offered only the services activated by default. Only the Linux honeypot was running a DNS server for name resolution in the local network; this server did not communicate with name servers outside the honeynet. The honeynet was then connected permanently to the internet without generating any outgoing traffic.

In **phase 2** the goal was to actively announce the existence of the honeynet in the internet. To achieve this, the host names were registered in the DNS servers of the university computing center and the zone transfer was activated in the local DNS server of the honeynet. In addition, a realistic domain name was registered for the honeynet. However, no specific services were offered by the honeypots and no outgoing traffic was generated.

In **phase 3**, we wanted to find out if the services offered by the honeypots would influence the attack patterns. In this phase we installed and activated commonly used services on the honeypots. Still, we didn't actively generate outgoing traffic.

In **phase 4**, we installed Peer-to-Peer (P2P) file sharing applications (KazaaLite) on two of the Windows machines to evaluate if active participation in file sharing networks would have any impact on the attack patterns. To stimulate access to our honeypots, we provided some content for download. In order not to violate any copyright laws, we fabricated fake content by generating files of a predefined size filled with random numbers. These files were then converted to valid MP3 files by using the LAME (http://lame.sourceforge.net) MP3 encoder and named according to current top10 hits.

5 Results of the Case Study

In our case study each of the phases had a duration of three weeks. After completion of a phase, the recorded data was statistically evaluated. This analysis was mainly based on the SNORT log-files. SNORT classifies attack signatures into severity levels. In the following we distinguish between:

- **Alarm:** dangerous and harmful attacks (SNORT priority 1)
- **Warning:** suspicious signatures potentially preparing attacks (priority 2)
- **Notice:** unusual traffic not identified as dangerous (priority 3)

5.1 Attacking Frequency

Since the first day the honeynet was running, activity could be detected confirming the statement that a honeynet will be found and attacked without further actions needed and that no significant "warmup" phase is required before starting statistical measurements. The honeypots have been scanned, probed or attacked every day with fluctuating intensity. Fig. 2 shows a typical summary of the recorded incidents. There is no obvious correlation between the intensity of alarms and warnings indicating the majority of attacks are blind, single phase attacks carried out without first scanning and fingerprinting the target (which would generate correlated warnings). One of our assumptions was that the attacking frequency would increase with the uptime of the honeynet because it becomes more widely known. However the measurements show that this is not the case and that external factors (malware activity) clearly define the attack frequency. Outbreaks of worm activities reported during the study could clearly be correlated to the measured attack intensity.

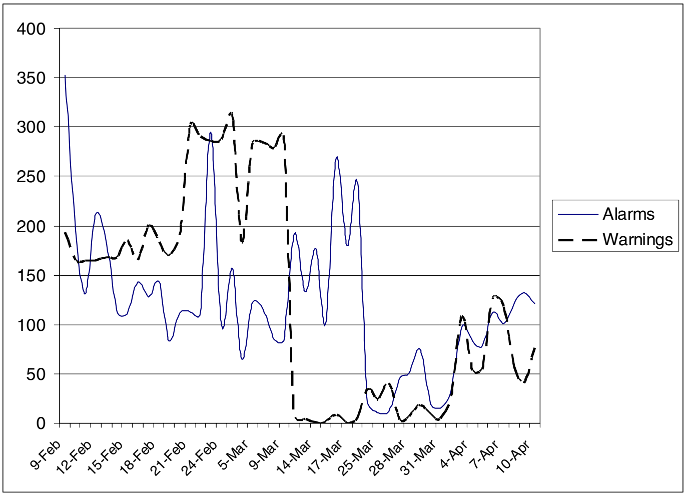


Fig. 2. Typical summary of the honeynet measurements from Feb. to Apr. 2004

5.2 Differences per Operating System

In the next steps of the analysis the distribution of attacks among the honeypots was evaluated. Fig. 3 shows an aggregation of all attack signatures (priority 1-3) detected over the complete study period and their allocation to the honeypots. In each of the

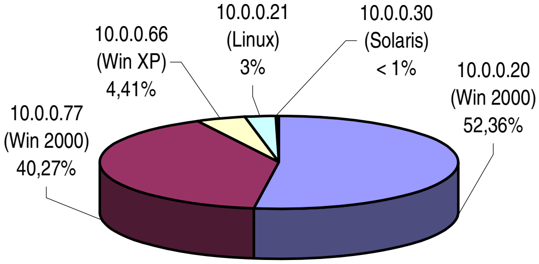


Fig. 3. Percentage of attack signatures per operating system[2]

[2] We used a public Class-C network for our honeynet, whose address we do not publish here for security reasons.

phases the Win2000 hosts were the primary targets of attack signatures, followed by the WinXP host. In all phases there were significantly less incidents reported for the UNIX systems Linux and Solaris.

The fact that nearly 97% of all signatures are targeted towards Windows systems was not unexpected since it makes sense to concentrate the effort to develop attacks on the clearly dominating operating systems. However, the conclusion that windows systems are significantly more insecure cannot be derived from these measurements as will be shown in the following.

5.3 Classification of Attack Signatures

We found a significant difference in the number of attack signatures per attack source (IP address of attacking host) between the Windows systems (average of 1) and the UNIX systems (average of 3). This led to the assumption that Windows attacks tend to be more blindly executed without first probing the target to prepare the attack. Since multi-phase attacks are more difficult to implement, single-phase attacks are probably automated to a larger extent. To validate this assumption the detected attack signatures were analyzed in more detail. By using the information provided by the "SANS Internet Storm Center" [http://isc.sans.org] the attack signatures of several active internet worms, especially MS-Blaster, Randex and SLAMMER have been identified. As a result, 81.2% of all alarms could be identified as automated worm attacks against Windows systems as shown in Fig. 4.

Only 1925 alarm signatures (18.8% of all detected alarms) were not worm related and thus potentially involve active human interaction. Since all of the worms try to exploit the same well known system vulnerability of Windows, all of the worm attacks can be blocked by installing one software patch. Therefore, the vast majority of attacks on Windows systems can obviously countered with minimum effort.

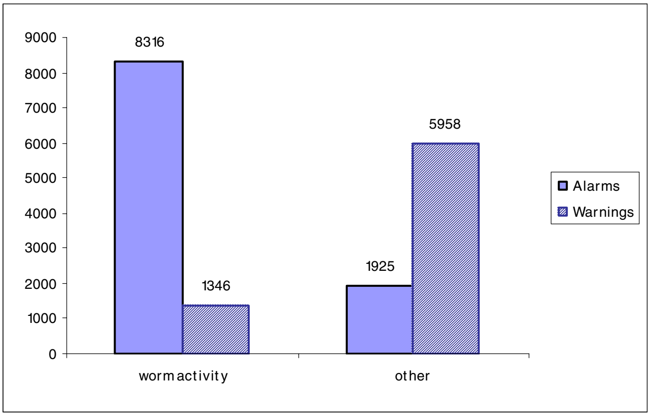


Fig. 4. Classification of attack signatures

For Linux systems, no specific worm activities could be identified. For the operation of a honeynet, the high number of identical worm attacks is only interesting for statistical purposes. Therefore it would make sense to automatically count and then filter out known worm signatures before generating the detailed reports, thus reducing the data volume and enabling the honeynet operator to concentrate on the more interesting attacks.

5.4 Impact of the Escalation Strategy on the Attack Frequency

One goal of the escalation study was to find out if there are methods to increase the attractivity of the honeynet and to attract more attacks. After eliminating attack frequency variations due to worm activity, the only configuration change deemed significant was the full activation of the DNS server. After starting the local DNS server and configuring the DNS reverse lookup on Dec. 16th 2003, the number of alarms increased significantly, as shown in Fig. 5, although no other configuration changes were made and no unusual waves of worm activity were reported. In particular, the DNS configuration caused a rise of the Microsoft specific RPC attack on port 135, so it can be assumed that this attack is correlated to DNS traffic.

The results of phase 4 were not quite as expected. The P2P search and download processes produced an enormous amount of data traffic, but no correlation could be detected between the P2P traffic and any attack signature. None of the IP addresses used in the P2P communication was involved in any non-P2P signature. Furthermore there was no temporal correlation between attacks and P2P traffic; also the overall attack frequency didn't increase significantly. From our results it can be concluded that making the honeynet known in the DNS is useful whereas actively generating traffic is not worthwhile and also clogs the log files with irrelevant data.

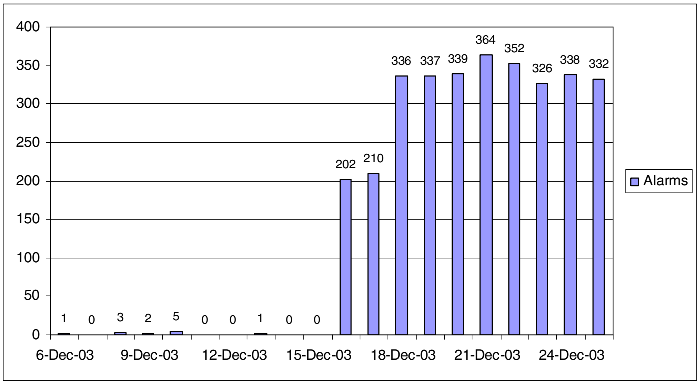


Fig. 5. Number of alarms in phase 2

5.5 Impact of the Escalation Strategy on the Attack Diversity

As shown in Table 1, the attacks generating alarms concentrated on the system services activated by default in the first two phases. In phase 3 when we activated http and ftp services, the variety of attacks increased significantly and these services became targets of specific attacks. However, the attacks were still unspecific in a sense that a significant part of the attacks on the web server were targeted towards the Microsoft IIS although only an Apache server was running. We also identified several attack signatures and a significant amount of attacks on popular services and applications not provided at all, e.g. SQL and SMTP servers. As a result, there seems to be a clear correlation between the variety of the popular services provided by the honeynet and the diversity of attack types.

Table 1. Number of different alarms per phase (grey fields mark provided services)

	Phase 1	**Phase 2**	**Phase 3**	**Phase 4**
Appl. not provided	-	-	2	2
HTTP	-	-	3	4
FTP	1	-	5	5
System services (act.)	3	2	4	5
Total	**4**	**2**	**14**	**16**

5.6 Usefulness for Forensic Analysis

In addition to statistical monitoring of attack activities, several distinct successful attacks were identified, observed and analyzed. In one example, one of the Win2000 honeypots was infected with the worm W32.Randex.Q. Besides trying to spread by automatically probing vulnerabilities on Microsoft's ports 135/tcp and 445/tcp, the worm installed a backdoor and automatically reported it to a remote server camouflaging the communication as IRC traffic (chat). Subsequently, various activities indicating human interaction were performed on the compromised system using different user names. Finally a program to send spam mails was installed and activated. Due to the data control mechanisms employed in the honeynet, both worm infection and spam distribution could be confined to the honeynet automatically.

Another attack specifically analyzed was a classical multi-phase attack. The attacker first scanned the network with ICMP packets to find active computers and then scanned on port 111/tcp to identify systems providing the RPC portmapper, which is typical for UNIX systems. In the third step the attacker did a standard RPC query for the "cachefsd" service on the Solaris honeypot to find out that this service is provided on port 32775/tcp. Subsequently the attacker launched a buffer overflow attack for the vulnerability known already since 2001. This proves that honeynets are efficient in a sense that non-automated (interesting) attacks can be observed in fairly regular intervals.

6 Conclusion and Outlook

In this paper we presented a practical case study of using a honeynet in a university environment. The study was performed over a period of several months to find out how efficient honeynets are for attracting, detecting, observing and documenting hacker activities within a computer network. In general, the honeynet concept proved to be quite useful, but also required a significant and continuing effort for setup, maintenance, supervision and analysis. The escalation strategy defined to identify factors which can increase the efficiency of a honeynet, i.e. the number and diversity of attack attempts per time period, led to the conclusion that the attack frequency is dominated by external factors, e.g. worm activity. However, by making the honeynet visible in the DNS system and by providing a comprehensive set of popular network services, the attractiveness of the honeynet can be increased. The active generation of traffic, e.g. by participating in P2P networks, however, seems to be counterproductive since it didn't attract more or more diverse attacks and made data analysis more difficult due to the massive amount of irrelevant data stored.

We are currently implementing a virtual honeynet [HON03b] in order to compare it to the classical setup with respect of efficiency and effort for implementation and maintenance. In addition we are considering mechanisms to filter out high volume, repeated and automated attacks to reduce the amount of stored data and to simplify the analysis of more interesting and divers attacks. Furthermore, we will continue our honeynet measurements in order to further investigate some effects we have observed.

References

[BER02] Bernstein, D.J.: Daemontools Homepage, http://cr.yp.to/daemontools.html

[ETH04] Official homepage for “Ethereal”, http://www.ethereal.com

[FLOYD00] A poor-man Tripwire-like system on Windows 9x/NT”
http://www.geocities. com/floydian_99/poormantripwire.html

[HON01] The Honeynet-Project: “Know Your Enemy: Revealing the security tools, tactics, and motives of the Black Hat community“, Indianapolis: Addison-Wesley, 2001, http://project.honeynet.org

[HON03a] Honeynet-Project: “Know Your Enemy: Honeynets”
http://www.honeynet.org/papers/honeynet/index.html

[HON03b] Honeynet-Project: “Know Your Enemy: Defining Virtual Honeynets”
http:// www.honeynet.org/papers/honeynet/index.html

[HON03c] Honeynet-Project: “Know Your Enemy: GenII-Honeynets”
http://www.honeynet.org/papers/gen2/

[HON04] Honeynet-Project: “Know Your Enemy: Honeynets in Universities”
http://www.honeynet.org/papers/edu/

[ICO02] Building a “sniffing cable” by IronComet Consulting,
http://www.ironcomet.com/sniffer.html

[LETH04] Lethi, Rami: “Advanced Intrusion Detection Environment”
http://source forge.net/projects/aide

[MOOR03] Moore, David: “The Spread of the Sapphire/Slammer Worm”
http:// www.cs. berkeley.edu/~nweaver/sapphire/

[NOR01] Northcutt, S.; Novak J.: “IDS: Intrusion Detection-Systeme“,Bonn: mitp-Verlag, 2001.

[NOR99] Northcutt, S.: “Network Intrusion Detection – An Analysts's Handbook”, Indianapolis: New Riders Publishing, 1999.

[OST04] Ostermann, Shawn: “ Tcptrace – Official Homepage”
http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html

[PRO03] Provos, Niels: “A Virtual Honeypot Framework”, CITI Technical Report, 01.10.2003, http://www.citi.umich.edu/techreports/reports/citi-tr-03-1.pdf

[ROE04] Roesch, Marty; Caswell, Brian: “Snort homepage”, http://www.snort.org/

[SIL03] Offizial homepage for “Snort Snarf“, http://www.silicondefense.com/software/snortsnarf/

[SPIT03a] Spitzner, Lance: “Open Source Honeypots – Learning with Honeyd” vom 20.01.2003, http://www.securityfocus.com/infocus/1659

[SPIT03b] Spitzner, Lance: “Open Source Honeypots, Part Two: Deploying Honeyd in the Wild” vom 12.03.2003, http://www.securityfocus.com/infocus/1675

[TCP04] Official homepage for tcdump: http://www.tcpdump.org/

[TRIP04] Official homepage for Tripwire Open Source, http://www.tripwire.org/

Measuring Round Trip Times to Determine the Distance Between WLAN Nodes[⋆]

André Günther and Christian Hoene

Telecommunication Networks Group (TKN), TU-Berlin, Germany
anguenther@gmx.de
hoene@ieee.org

Abstract. This publication explores the degree of accuracy to which the propagation delay of WLAN packets can be measured using today's commercial, inexpensive equipment. The aim is to determine the distance between two wireless nodes for location sensing applications. We conducted experiments in which we measured the time difference between sending a data packet and receiving the corresponding immediate acknowledgement. We found the propagation delays correlate closely with distance, having only a measurement error of a few meters. Furthermore, they are more precise than received signal strength indications.

To overcome the low time resolution of the given hardware timers, various statistical methods are applied, developed and analyzed. For example, we take advantage of drifting clocks to determine propagation delays that are forty times smaller than the clocks' quantization resolution. Our approach also determines the frequency offset between remote and local crystal clocks.

1 Introduction

Knowing the position of wireless nodes is required for location-aware services and applications. The position can be calculated using the distance between wireless nodes. Furthermore distance helps when deciding the time of handovers or finding the optimal routing path throughout an ad-hoc network.

In this paper we focus on locating techniques which use the intrinsic features of WIFI based wireless access. Usually, received signal strength indications are applied to identify the location of wireless nodes. We show that precise distance measurement based on round trip time measurements of WLAN packets is possible even with low-cost, commercial WLAN hardware. We developed the algorithms to determine the air propagation time indirectly and to improve the accuracy and resolution of the time measurements. We validated our approach with two independent experimental measurement campaigns and with an analytical explanation.

[⋆] This work has been supported by the Deutsche Forschungsgemeinschaft (DFG). This publication is a condensed and enhanced version of [1].

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 768–779, 2005.

We utilize the following standard feature of IEEE 802.11: Each unicast data packet is immediately acknowledged by its receiver (Fig. 1). We took the time between starting the transmission of a data packet and receiving the corresponding immediate acknowledgement. We will refer to this as remote delay (d_{remote}). We also measured the duration of receiving one data packet and sending out the immediate acknowledgement. We will call this duration local delay (d_{local}). The overall propagation time is then estimated by subtracting the local from the remote delay.

$$c = \frac{2 \cdot distance}{d_{remote} - d_{local}} \text{where } c \approx 3 \cdot 10^8 \frac{\text{m}}{\text{s}} \text{ being the speed of light.} \qquad (1)$$

In order to overcome the problem of interrupt latencies and hence inaccuracies when measuring the duration of packet transmission in the operating system, we measured the time on the hardware layer - the WLAN card. Most WLAN solutions allow to record time stamps at a resolution of 1 μs. However, a packet travels a distance of 300 m in 1 μs, which usually exceeds the range of WLAN transmission. We increase the resolution by using multiple delay observations and applying statistical methods to enhance the accuracy.

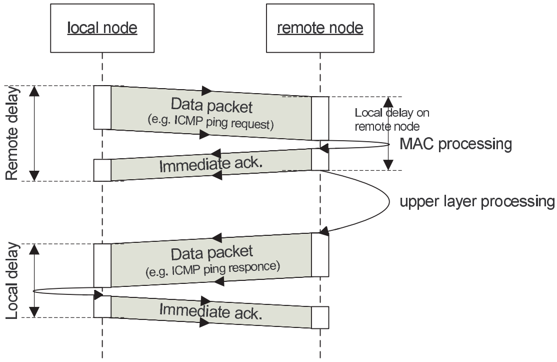


Fig. 1. Distance measurement: Transmission of an ICMP ping sequence

This paper is structured as follows: In Sect. 2 we refer to the state of the art. Then we explain our approaches to enhance the measurement resolution. In Sect. 4 we describe our experimental measurement campaigns. Finally, we briefly summarize the results and contributions of this paper.

2 Related Work

A couple of approaches to in- and outdoor location sensing techniques have been presented [2]. An essential part of location sensing algorithms is a method to determine the distance between two wireless nodes. In general, three methods have been considered. Firstly, in case of densely populated networks such as sensor networks [3] the information about which nodes are within transmission range is used. Secondly, the received signal strength indication (RSSI) of data packets transmitted is considered. It decreases sharply in a non-linear fashion with distance, so that environment specific signal strength maps relating RSSI values to positions have to be created first. An example for RSSI application is the RADAR system [4], which has been one of the first approaches presenting an indoor positioning system based on WLAN components (overview in [1]). Thirdly, the propagation time of radio signals can be used because in free air it linearly increases with distance. Such an approach is usually considered to be impossible without the help of special signal processing hardware [5].

The classic approach to the latter method of position location estimates the time of arrival (TOA) of pure radio signals (instead of WLAN packets). This is conducted by applying signal processing algorithms based on cross-correlation techniques [6]. The TOA method suffers from multi-path conditions. This problem can be encountered with a wider frequency band, e.g. ultra-wide band.

TOA measurement is being employed both outdoors for GPS-positioning [7] and indoors to find things and people marked by a tag [8]. In the latter paper, the author gives an appraisal of the achievable accuracy when measuring the round trip TOA within the 2.44 GHz and 5.78 GHz bands. For a signal bandwidth of 40 MHz, the accuracy of 3.8 m can be an achievable resolution limit unless further signal processing techniques are applied. Those might enhance the resolution up to 1 m.

The only paper focussing on measuring pure packet propagation delays is [9]. The objective is to determine the speed of light using the averaged measured round trip propagation delay of ping packets. The measurements were conducted in a wired Ethernet infrastructure. Estimating the propagation delay which ranges below the clock resolution was facilitated by employing the concept of noise-assisted sub-threshold signal detection. For measurements in an IEEE 802.11b wireless environment the round trip times were too variable and noisy to be used.

3 Approach

Inspired by the approach presented in [9] we also use the mean round trip time delay of packets to determine the distance as given in (1). In order to keep the time measurements as unbiased as possible resulting in a high resolution, we try to preclude any disruption caused by operating system activities. To do so, we took the following action:

Firstly, we utilized the IEEE 802.11 data/acknowledgement sequence instead of the ICMP-Ping request/response packet sequence. As the ping response is generated by the operation system the time it takes is subject to a highly variable delay. In contrast, the immediate acknowledgements are handled by the hardware of the WLAN radio and hence highly predictable. As for our measurements we assumed the MAC processing time to be equal on both wireless nodes. Although the MAC processing time is standardized according to IEEE 802.11 we will prove that not all WLAN cards operate in compliance with the standard. In practice, the MAC processing time also depends on the chip set hardware and firmware of the actual WLAN cards in use. To account for this a model-specific absolute delay offset needs to be considered.

Secondly, we do not measure the time stamps for packet arrival and transmission on the operating system layer, but on the WLAN card hardware layer. This features measuring conditions that are independent of variable interrupt latencies. In [10] we showed that measuring the time of a packet's arrival in the operating system's kernel (e.g. during an interrupt) entails quite imprecise results due to falsification by the variable interrupt latency.

The resolution of these hardware time stamps, which are implemented in most current WLAN products, is 1 μs corresponding to 300 m. In terms of the achievable accuracy this discrete time resolution is not precise enough yet. The resolution increases when averaging numerous observations. In the following we consider three phenomena that help to achieve a higher resolution.

Gaussian Noise: The presence of measurement noise is assumed. Noise can be caused by thermal noise in the received radio signal or by the presence of multipath environment. Also, the crystal clocks of the WLAN equipment are subject to a constant clock drift and variable clock noise. Thus, the delay values are not limited to only one value. (In Fig. 2 not only 323 μs can be observed but also other values). If one assumes a Gaussian noise distribution with a suitable strength, we can simply take the sample mean to enhance the resolution.

Stochastic Resonance: Instead of the explanation above the authors of [9] suggested another statistic effect called stochastic resonance. The concept of stochastic resonance was originally introduced as an explanation for the periodically recurrent ice ages. In the last two decades, it has been applied to explain many physical phenomena [11]. In the realm of signal detecting stochastic resonance allows for detecting signals below the resolution of the measuring units because the signal becomes detectable with the help of noise. Noise adds to the signal so that it eventually exceeds the threshold given by the resolution of the detecting device. Thus, the system is able to change its states. The state durations have random lengths, but the probability is high that one state remains the same in the next observation.

Beat Frequencies: In our experiments (Fig. 4.3 in [1]) it can be observed that the 323 and 324 values occur in blocks of regular patterns. But this effect cannot be explained with the effect of stochastic resonance. Another effect can also entail resolution enhancement even if measurement noise is missing: 'Relative clock drift' – both WLAN cards are driven by built-in crystal oscillators that have nearly the same frequency. Due to tolerances, there is a slight drift between both clocks which causes varying rounding errors.

Let us consider the impact of a discrete time resolution on the measurement error. Firstly, we construct a model of the experiment setups. Instead of using packets, we assume that a delta pulse is sent off from the local to the remote node. After the delta pulse's arrival another delta pulse is sent back to the local node representing an acknowledgement. The local node can only process the impulses only in discrete time steps $t_{local} \in \mathbf{N}$ described with natural numbers. The same is also valid for the remote node. It only reacts in discrete time steps, which are $t_{remote} = \delta + n$ where $n \in \mathbf{N}$ and phase offset of $\delta \in [0; 1[$. We assume that the clocks work at the same speed but with a phase offset. Moreover, another assumption is that phase offset changes over time but not for the duration of a round trip. The transmission of a delta impulse from one node to the other takes the delay of $d_{prop} \in \mathbf{R}^{+}$, which is equal to the propagation time.

Let us assume that a delta impulse is sent off from the local node at the time t_{local}^{out}. It arrives the remote node after a period of d_{prop}. Due to the discrete MAC processing, the delta impulse is only identified at the next remote clock impulse, which is:

$$t_{remote}^{in} = \left\lceil \left(t_{local}^{out} + d_{prop}\right) - \delta \right\rceil + \delta \tag{2}$$

Assuming a MAC processing duration equal to zero and $t_{remote}^{out} = t_{remote}^{in}$, the remote node immediately sends back a delta impulse representing the acknowledgement. It arrives at the local node after a period of d_{prop}, but is again only recognized at the next local clock, which is

$$t_{local}^{in} = \left\lceil t_{remote}^{out} + d_{prop} \right\rceil \tag{3}$$

Then, the observed round trip time *rtt* is (4).

$$\begin{aligned} rtt = t_{local}^{in} - t_{local}^{out} &= \lceil \lceil t_{local}^{out} + d_{prop} - \delta \rceil + \delta + d_{prop} \rceil - t_{local}^{out} \\ &= \lceil d_{prop} + \delta \rceil + \lceil d_{prop} - \delta \rceil \end{aligned} \tag{4}$$

Next, we assume that the phase changes from one to the next measurement. The change is constant and is repeated after each phase period starting at zero again. In the following, we only consider one phase period and assume that round trip times are measured at all times. Thus, the number of observations is infinite. The mean *rtt* over all phase offsets is calculated as follows.

$$\overline{rtt} = \int_0^1 rtt\, d\delta = \int_0^1 \lceil d_{prop} + \delta \rceil + \lceil d_{prop} - \delta \rceil \; d\delta = \; 2 \cdot d_{prop} + 1 \tag{5}$$

The variance of the quantization error is calculated as followed and is simplified to a cubic function of the fractional part of the round trip distance. Both the mean and variance are displayed in Fig. 3.

$$\sigma^2 = \int_0^1 \left(\overline{rtt} - rtt\right)^2 \; d\delta = \{2d_{prop}\} - \{2d_{prop}\}^2 = \tfrac{1}{4} - \left(\{2d_{prop}\} - \tfrac{1}{2}\right)^2 \tag{6}$$

The *rtt* function produces a pattern which is repeated every phase period. This reoccurrence introduces a frequency component to be present in the observations. If two clocks interfere, their phases are equal every beat period, which is the reciprocal of the beat frequency. The beat frequency is the difference of both frequencies of both interfering waves (7).Thus, the impact of quantization errors causes a similar effect as the two interfering waves – namely a beat frequency.

$$f_{beat} = |f_1 - f_2| \tag{7}$$

Limits and Verification: The accuracy of location and distance sensing algorithms have fundamental limits (refer to the citations in [1]). For example, the analytic calculations above do not take into account the clock drift during one RTT observation. Assuming a frequency stability of ±25 ppm and a length of a transmission sequence of 60 μs and 320 μs, the maximal error could be up to 0.9 m and 4.8 m respectively.

Furthermore, one should note that only in vacuum light travels at the speed of light c. In materials the propagation speed depends on the square root of the dielectric constant ε. For example, dry ferroconcrete has an ε of about 9 and electromagnetic waves traverse through ferroconcrete 3 times slower than in vacuum. Most other materials used in buildings have lower dielectric constants.

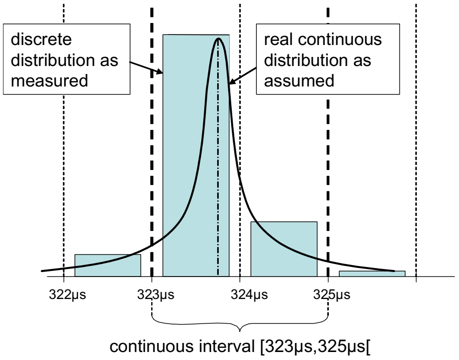


Fig. 2. Discrete distribution of noisy delay measurements

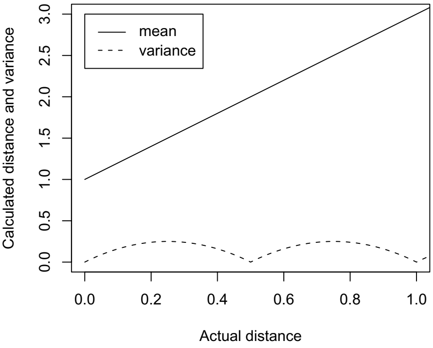


Fig. 3. Theoretical mean distance and variance of distance

Another source of possible errors is due to non-line-of-sight conditions. This results in an overestimation of the distance between the two nodes [12]. Multi-path propagation might introduce measurement errors because the dominant path can vary depending on the current transmission conditions. Multi-path propagation is only present if reflections are given. Reflections can have large impact on signal strength but only a low one on propagation delay. Thus, in the presence of multi-path propagation or reflections, we assume time delay measurement as being more precise than those based on the RSSI.

In order to check these hypotheses and identify the real measurement resolution, we conducted experiments. The first measurement campaign was conducted to study the impact of slow-user motion on packet loss and delay as described in [10]. At the same time, we also measured the impact of distance on the round trip times. One year later, we embarked on a second measurement campaign. We altered the radio modem technology, the location, the analysis software, and the staff. The consistence of both results proves the reliability and correctness of our approach.

4 Measurements: First and Second Campaign

Experimental Setup: The measurement was conducted twice: First in a gymnasium [10] and later in the in the countryside where one could expect the channel to be free of disturbing noise coming from other radiating devices. The data communication took place between the local and the remote node. ICMP ping packets were transmitted each 20 ms respective 10 ms. The measurements of RTT were conducted for several distances: First covering the range from 5 to 40 m, later extend to the maximal transmission range of 100 m.

At each distance, we measured for about 15 minutes respective 4 minutes. One should note, that in this first campaign, the wireless LAN cards were situated close to the ground. Also, the directions of the antennas were selected at random and were not recorded. This is important to know as it explains some of the results presented later. In the second session the sender was placed on a plastic table, whereas the receiver was installed on top of a 1.5 m wood-metal ladder. This was to guarantee that a large percentage of the Fresnel-zone, an elliptic space around the direct line-of-sight between both nodes is free of any obstacles harming the transmission. This time, the antennas were directed toward each other.

Equipment: The PCs were running a Suse 6.4 Linux system with a 2.4.17 kernel (A). D-Link cards featuring an Intersil's (now Conexant) Prism2 chipset were employed as a wireless interface. Packets were directly sniffed on the MAC layer by the measurement tool 'Snuffle'.

The second time, we used an access point (Netgear FWAG114) supporting 802.11b/g as remote node. The PCs were running under Linux, Suse 9.1, with a special 2.6 kernel. We used two different WLAN cards containing chip sets from Atheros and Conexant implementing IEEE 802.11 a,b and g. The Atheros cards (brand Netgear WAG-511, contained an AR5212 chip) are supported by the Madwifi device driver. We used the software version downloaded from the CVS server on the August 30^{th}, 2004. The Conexant cards (brand: Longshine LCS-8531G containing Prism-GT chipset with an ISL3890 as MAC-Controller) are controlled by the prism54.org device driver (date 28-06-2004, firmware 1.0.4.3.arm). During each measurement both the sender and monitor were equipped with cards of the same brand. To gather the packet traces, we used tcpdump and libpcap.

Configuration: WLAN networking technologies based on the IEEE 802.11 standards transmit data packets via air. To avoid potential packet delay effects, in the first experiments the maximal number of retransmissions (transmission type) was set to zero. The second measurements were conducted in seven different configurations to study the impact of the WLAN card, CPU clock and modulation type. We used the default configuration of WLAN cards and access point but changed the supported standard to 802.11g and set the modulation type to either 36 or 54 Mbit/s. The frame length of the data packets are 65 bytes and of the acknowledgements 14 bytes.

Time Measurements: All three different WLAN cards recorded the arrival time of packets at a resolution of 1 μs without any variable latency. The precise point of time, at which the time stamp is recorded, is not documented. Also, the WLAN chip sets feature only the recording of time stamps of incoming packets. But we needed both sending and receiving time stamps. Therefore, we decided to use a third PC to monitor the packets which the local node sends and receives. The monitor PC was placed close-by the sender to avoid any additional propagation delays that could falsify the measurements.

It will be straight forward to alter software and firmware of WLAN cards to record transmission time stamps, too. Due to legal constraints, we were not able to implement these changes by ourselves. We expect that WLAN chipset manufactures will provide firmware updates to support precise time stamps because they will benefit from customers using WLAN for location-aware services. Until then, we are required to use the third monitoring node.

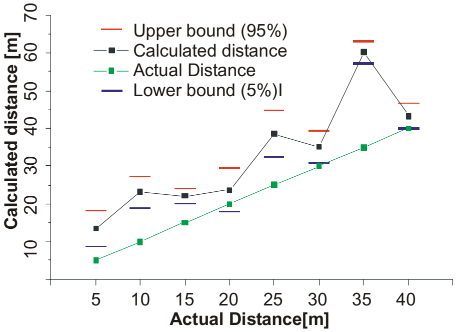
Data Collection & Processing: Snuffle provides the packet traces of all 802.11b packets received at the monitoring node. We filtered-out only the successful ping sequences which consist of an ICMP request, an acknowledgement, an ICMP response and again an acknowledgement. Other packets like erroneous transmissions, beacons, ARQ messages etc. were dropped. Due to hardware limitations of the WLAN card only a fraction of observations were recorded.

Only the delays fitting in the interval [323 μs, 324 μs] are considered in further calculations (Fig. 2). A few delay measurements were observed with the value of 322 and 325 μs. These and all other delays were considered as measurement errors. Taken the valid packet sequences, the mean and variance of the remote delay and local delay were calculated. To check for stationary process properties, the autocorrelation function was calculated.

In the second round, Tcpdump recorded the packet traces and wrote them to files. After the measurements we used tcpdump to convert these files to plain text files. Tcpdump had to be modified in order to print out the prism link-layer headers. For statistical analysis the R project software turned out to be quite efficient. Thus, this time we applied R programs to calculate the data's analyzed mean, variance and autocorrelation.

Results: The distance was directly derived from the measured propagation delay using equation (1). Assuming a Gaussian error distribution, we also plotted the confidence intervals in Fig. 4. In the first campaign the calculated distances were always higher than the real distances. Also, in some measurements (e.g. 35 m) the air propagation time was significantly higher. Due to the experimental setup, we could not ensure that the direct line-of-sight path was taken. The remote node was placed directly on the ground. Thus, the Fresnel zone was violated and the direct transmission path was hampered.

In Fig. 5 the signal strength is displayed as a function of the distance. Theoretically, the signal strength should decrease with distance. In this measurement campaign other factors, such as reflection, seem to be dominant. If one compares Fig. 4 and Fig. 5, it seems time measurements reflect the distance more precisely than RSSI but they have a higher variance and a larger confidence interval. The



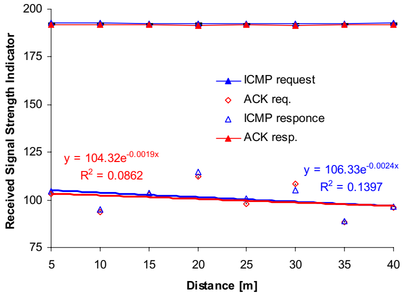


Fig. 4. Distance as calculated from RTT versus actual distance between both nodes. 95% confidence levels are given

Fig. 5. Received signal strength indication versus distance. Confidence intervals are too small to be shown

results of the second round are illustrated in Fig. 6, which shows the remote (blue) and local delay (red) measurements, the number of overall observations (#) and the correlation coefficient (R) for the given configuration. A clear correlation between actual distance and calculated distance can be identified. In the right graph, one can see that the larger the distance (and the worse the link quality), the larger the confidence interval becomes. In Fig. 9 we display the variance of *rtt* observations over the distance. The curve is highly similar to the curve described with (6). Thus, our beat-frequency explanation seems to be valid.

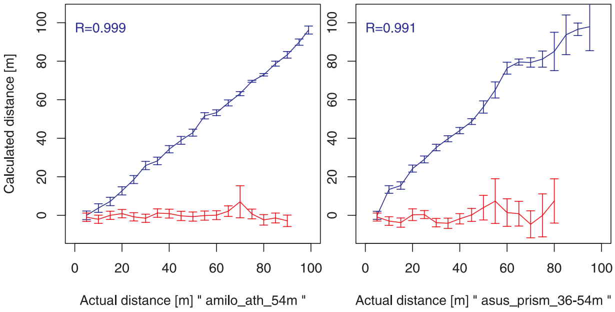


Fig. 6. Propagation delay (=calculated distance) vs. actual distance (plus 95% conf. intervals). (blue/upper lines=biased remote delay, red/lower lines=biased local delays). Each value is based on at least 1000 observations

Analysis: In [1] we show that the first measurements follow a weak stationary process, with a constant mean, variance and covariance (for a constant lag). Thus, further statistical methods are applicable: Confidence intervals are only meaningful if the observations are independent. This assumption can be verified by the autocorrelation function. The time-lag dependent autocorrelation coefficients are presented as a graph in Fig. 7. The 40 m results are shown as an example. The autocorrelation for the local delay is low. It is smaller than ρ=0.05. Thus, the local delay measurements can be seen as independent. The autocorrelation of remote delay values

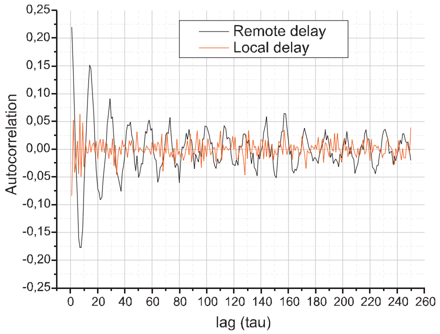


Fig. 7. Autocorrelation (=cross correlation of itself) is oscillating for remote delays – indicating a fundamental frequency component in observations (at 40 m)

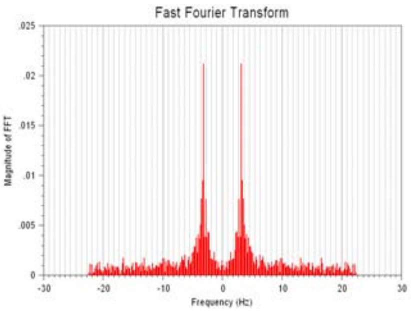


Fig. 8. The Fourier transformation of the observations shows a dominant frequency at 3.5 Hz, which is only present in the remote delays. (at 40 m)

has the shape of a decaying cosines wave. This kind of autocorrelation curve is found if the observations feature a constant frequency component. Indeed, this pattern arises in the delay traces. The values of 323 and 324 occur block-wise in bursts. We also calculated an FFT over the packet delays. Assuming that each observation follows the previous after 20 ms, we identified a dominant frequency of about 3.5 Hz independent of the distance (Fig. 8). However, the lower the packet error rate, the stronger this effect is. We also calculated the autocorrelation of the second measurement's second results, high and alternating correlation coefficients were only present, if we used the Prism GT chip sets. We assume that this observation is due to the clocking of the MAC protocol and due to the frequency stability and accuracy of the WLAN quartz crystals. Further studies are required to understand this effect in-depth.

We explain the effect displayed in Fig. 7 with interference of both remote and local crystal clocks. Taken this explanation of quantization errors we can calculate the clock drift between both signals. Assuming a clocking of the MAC protocol at 1 MHz, the drift between both clocks is approximately $drift = \frac{f_{beat}}{f_1} = \frac{3.5Hz}{1MHz} = 3.5ppm$. Usually, the tolerance of consumer grade quartz clocks is up to 25 ppm. Thus, we consider this explanation to be plausible.

Interestingly, the MAC processing is conducted in steps of 1 μs. Thus, the MAC processing time is not precisely the SIFS interval but is rounded up to the next 1 μs. However, the error is small so that receivers tolerate it.

In our quantization error analysis we calculated the variance which is up to $1/4$. A distance of one and a time unit of one in the analysis refer to 300 m or 1 μs in the experiments. Then, the standard deviation would be 18.75 m or 62.5 ns at most. The measured standard deviation ranges between 3.3 and 25 m. Thus, the quantization error is not the only dominant effect and others such as thermal noise are important too.

We measured at each distance for 4 to 15 minutes. Is it really required to measure that long? In Fig. 10 we consider only a subset of all *rtt* observations

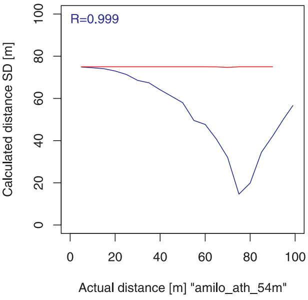


Fig. 9. The variance of *rtt* observations over time

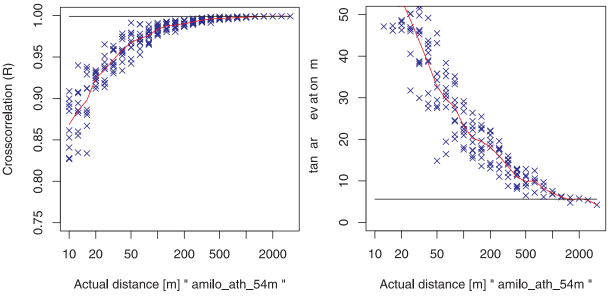


Fig. 10. The accuracy (cross correlation and standard error) over the number of observations per position

taken during the second campaign. We display the correlation coefficient R and the standard error over the number of observations per distance. With 500 to 1000 observations per position nearly the optimal accuracy is achieved. If one assumes that a packet is sent off every mircosecond, the distance can be estimated after 1 s of continues transmission.

5 Conclusion

We have presented an algorithm to measure the air propagation time of IEEE 802.11 packets with a higher accuracy. Using two different experimental setups, we determined the precision of round trip time measurements. We used commercial WLAN cards, supporting IEEE 802.11b and 802.11g, implemented with three different WIFI chip sets. We have shown that such time measurements are possible even with off-the-shelf, commercial WLAN equipment and without additional signal processing hardware.

To overcome the low resolution of the clocks, numerous observations have to be combined and smoothened. This can be carried out best during an ongoing data transmission at no additional cost. We explained why smoothing indeed helps to enhance the resolution of the time difference measurement so that distance measurements become possible. This effect can be due to the presence of measurement noise and to the beat frequency resulting from drifting clocks. To the best of our knowledge, especially the latter explanation is novel.

Our finding suggests that instead of RSSI the round trip time should be measured because it is correlated with the distance more strongly. In our gymnasium measurement the RSSI has not been useful to identify the distance because – due to reflections – the attenuation varied largely.

The contribution of this work is to show that neither synchronized, precise clocks nor special hardware is required if the propagation delay between two WLAN nodes is to be measured. This allows the implementation of easy-to-use, cheap and precise indoor positioning systems, which do not require maps containing signal strength distributions. However, WLAN chipset manufacturers should update their firmware so that it reports the round trip time of packets with an accuracy of at least 1 μs. Then, a 1000 packets transmission – achievable in less than one second – can measure the distance with an error deviation of less than 8 m.

Acknowledgements

We like to thank Prof. Wolisz for his ongoing support, E.-L. Hoene for the revision, and Sven Lamprecht and David Hundenborn for conducting the second measurements.

References

1. Günther, A., Hoene, C.: Measuring round trip times to determine the distance between WLAN nodes. Technical Report TKN-04-016, Telecommunication Networks Group, Technische Universität Berlin (2004)
2. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. IEEE Computer **34** (2001) 57–66
3. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: Proceedings of MOBICOM, San Diego, CA, ACM Press (2003) 81–95
4. Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-Based User Location and Tracking System. In: Infocom 2000, Tel-Aviv, Israel (2000) 775–784
5. Velayos, H., Karlsson, G.: Limitations in range estimation for wireless LAN. In: Proc. 1st Workshop on Positioning, Navigation and Communication (WPNC'04), Hannover, Germany (2004)
6. Alsindi, N., Li, X., Pahlavan, K.: Performance of TOA estimation algorithms in different indoor multipath conditions. In: IEEE Wireless Communications and Networking Conference (WCNC). Volume 1. (2004) 495–500
7. Enge, P., Misra, P., eds.: Special issue on GPS: The Global Positioning System, IEEE (1999)
8. Werb, J., Lanzl, C.: Designing a positioning system for finding things and people indoors. IEEE Spectrum **35** (1998) 71–78
9. Lepak, J., Crescimanno, M.: Speed of light measurement using ping. American Physical Society - Meeting Abstracts (2002) abstract B2.009.
10. Hoene, C., Günther, A., Wolisz, A.: Measuring the impact of slow user motion on packet loss and delay over IEEE 802.11b wireless links. In: Proc. of Workshop on Wireless Local Networks (WLN) 2003, Bonn, Germany (2003)
11. Gammaitoni, L., Hanggi, P., Jung, P., Marchesoni, F.: Stochastic resonance. Reviews of Modern Physics **70** (1998) 223–287
12. Cong, L., Zhuang, W.: Non-line-of-sight error mitigation in mobile location. In: Infocom 2004, Hong Kong (2004) 650– 659

End-to-End Asymmetric Link Capacity Estimation*

Ling-Jyh Chen, Tony Sun, Guang Yang, M.Y. Sanadidi, and Mario Gerla

Computer Science Department, UCLA, Los Angeles, CA 90095, USA
{cclljj, tonysun, yangg, medy, gerla}@cs.ucla.edu

Abstract. Knowledge of link capacity is important for network design, management, and utilization. With the increasing popularity of asymmetric link technologies (such as DSL, 1xRTT, and satellite links), it is desirable to have a capacity estimation technique, which can simultaneously measure forward and backward direction link capacities on an Internet path. Moreover, this estimation must often be "sender only", because of receiver limitations or lack of standards. In this study, we propose a simple, fast and accurate technique, called AsymProbe, to estimate asymmetric link capacities. AsymProbe is a "sender only", round trip procedure. It achieves asymmetric link capacity estimation by strategically altering the ratio of probe and acknowledgement packet sizes. Using simulation and testbed experiments, we validate AsymProbe with a variety of network configurations. The results show that AsymProbe can correctly estimate the asymmetric link capacities as long as an appropriate packet size ratio can be employed.

1 Introduction

Knowledge of link capacity is particularly important for network design, management and utilization. A simple and accurate scheme for capacity measurement and monitoring is becoming increasingly desirable, especially for emerging technologies and applications such as overlay, peer-to-peer (P2P), sensor, grid and mobile networks. A successful capacity estimation solution will need to encompass speed of execution, simplicity, accuracy, and extendibility beyond the limits of traditional networks, in particular the increasingly popular *asymmetric* access methods to the Internet, e.g. DSL, cable modem and satellite links. It is also often imperative to carry out the estimation in a round trip, "sender only" fashion. This is because the receiver is not powerful enough to implement the estimation algorithm. It must, however, participate in the response to probe packets.

Several capacity estimation methods exist, including CapProbe [4], which is sender only and fast. However, sender only methods so far have addressed symmetric path extimations, ie, the minimum capacity is the same in both directions.

* This material is based upon work supported by the National Science Foundation under Grant No. CNS-0435515.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 780–791, 2005.

Yet, asymmetric links do exist; moreover, many applications are intrinsically asymmetric too and thus can benefit from the knowledge of such asymmetries. For example, in multimedia streaming and file downloading, bulk data is transmitted only in the forward direction, consuming much more bandwidth than the control traffic in the reverse direction. In this case, the knowledge of one-way capacity on the *forward* direction link is mandatory, as it is a much better predictor of the streaming or downloading rate, than the blindly measured round-trip bottleneck capacity, accounting for the cases when the forward link has larger capacity than the backward link.

Previous approaches on capacity estimation can be divided into two categories: one-way probing (e.g. Pathrate [1]) and round-trip probing (e.g. CapProbe [4]). In [4], a thorough comparison of modern capacity estimation methods was presented, where CapProbe was especially singled out as a fast and accurate capacity estimation mechanism addressing both wired and wireless links. However, limited by its round-trip nature, CapProbe only works well on symmetric links. When operating on an asymmetric link, CapProbe measures the narrower capacity of the two directions. It cannot distinguish the respective capacities of the forward and backward links.

Even though capacity estimation for asymmetric links can be achieved by conducting single direction capacity probing (e.g. Pathrate) for the two directions separately, this estimation strategy is often considered undesirable, as it imposes unnecessary computation overhead and complexity on the receiver (eg, the mobile host). Moreover, it requires compatible software and consistent computation methods in both hosts. To simplify the process of estimating asymmetric link capacities, round trip capacity probing is still the most desirable solution. Still, existing method like CapProbe, lacked such a capacility, modifications are needed to add support for accurate capacity estimations of asymmetric links.

To this end, in this study, we propose and evaluate a round trip technique for estimating asymmetric link capacities called AsymProbe. AsymProbe is engineered based on the well proven CapProbe mechanism. Through careful selection of probe and acknowledgement packet sizes, AsymProbe can successfully provide simple, fast, and accurate capacity estimates for asymmetric links.

The rest of the paper is organized as follows. In section 2, we survey and summarize work related to this study. An in-depth description of AsymProbe follows in Section 3. In section 4, we evaluate the accuracy of AsymProbe in estimating link capacity through series of NS2 simulations. In section 5, we present results from our testbed experiments to validate the capability of AsymProbe. Section 6 concludes the paper.

2 Background and Related Work

Previous research on capacity estimation relied on either delay variations among probe packets as illustrated in pathchar [3], or dispersion among probe packets as described in Nettimer [6] and Pathrate [1]. The analysis in [1] clearly revealed that the dispersions distribution can be multi-modal without multi-channels,

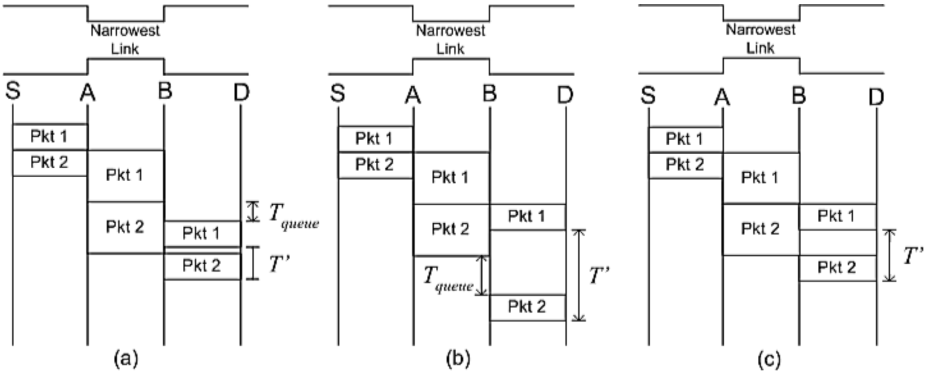


Fig. 1. (a) under-estimation caused by "expansion" (b) over-estimation caused by "compression" (c) the ideal case. (T': Measured dispersion; T_{queue}: Queueing delay)

and that the strongest mode in the multimodal distribution of the dispersion may correspond to either (1) the capacity of the path, or (2) a "compressed" dispersion, resulting in capacity over-estimation, or (3) to the Average Dispersion Rate (ADR), which is lower than the capacity.

Other tools such as pchar and clink [2] use variations of the same idea as pathchar. Pchar employs regression techniques to determine the slope of the minimum RTT versus the probing packet size. However, pathchar-like tools have limitations with respect to the speed of estimation process as shown in [4].

CapProbe [4] is a recently proposed capacity estimation technique shown to be both fast and accurate over a large range of scenarios. When a back-to-back packet pair is launched into a network, it is always dispersed at the bottleneck link according to the bottleneck capacity. If such dispersion is preserved until the pair arrives to destination, it identifies the bottleneck capacity (as shown in Fig. 1-c). Unfortunately, the dispersion can be either expanded or compressed, where "expansion" of dispersion leads to under-estimation and "compression" of dispersion leads to over-estimation of the capacity, as shown in Fig. 1-a,b.

To overcome this problem, CapProbe combines the use of dispersion and end-to-end delay measurements thus filtering out packet pair samples distorted by cross traffic. Whenever an incorrect value of capacity is estimated, either the first or the second packet, or both, have been delayed by cross traffic. In this case, the sum of the delays of the two packets in the packet pair, called the delay sum, includes some queuing delay. A delay sum that does not include any queuing delay introduced by cross traffic is referred to as the minimum delay sum. The dispersion of such a packet pair sample is not distorted by cross traffic and reflects the actual capacity. A valid sample can easily be identified since its delay sum is the minimum among delay sums of all packet pair samples. The capacity is then estimated by the equation:

$$C = \frac{P}{T} \tag{1}$$

where P is the sampling packet size, and T is the dispersion of the sample packet pair with the minimum delay sum.

The majority of the existing capacity estimation tools, including the ones discussed above, are inherently round-trip based. They estimate the narrowest capacity on the round-trip path. These techniques encounter severe constraints when measuring link capacities of increasingly popular asymmetric links, such as DSL, cable modem and satellite links where the forward link capacity is very different from the backward link capacity. In this study, we propose AsymProbe, a novel scheme that measures asymmetric link capacities in the round trip fashion. Details of this proposed approach will be presented in the following sections; evaluation of AsymProbe will be discussed in the simulation and experiments sections.

3 Proposed Approach: AsymProbe

In this section, we present AsymProbe, a novel capacity measuring technique that allows to measures the capacity of either the forward or backward narrow link on the path. The basic idea of AsymProbe stems from the observation that the measured dispersion in the original CapProbe can be introduced either in the forward or backward direction of an asymmetric link. When probing and acknowledgement packets are of same size, the measured dispersion is good for estimating the round-trip bottleneck capacity, since the narrowest link along the round-trip path gives the largest dispersion to the (probing or acknowledgement) packet pairs. One can then easily estimate this capacity by applying Eq. 1.

Fig. 2 depicts the packet pair interactions in an asymmetric link scenario, with link capacity C_1 on the forward direction link and capacity C_2 on the backward direction link. The probe packets are sent back-to-back with packet size P_1 on the forward direction link (from A to B); the acknowledgement packets are sent immediately upon receipt of probe packets with packet size P_2 on the backward direction link (from B to A). Suppose T_1 and T_2 represent the respective dispersions of probe packets and acknowledgement packets when they are sent back-to-back on the link; from the definition of Eq. 1, T_1 and T_2 can then be derived as $T_1 = \frac{P_1}{C_1}$ and $T_2 = \frac{P_2}{C_2}$.

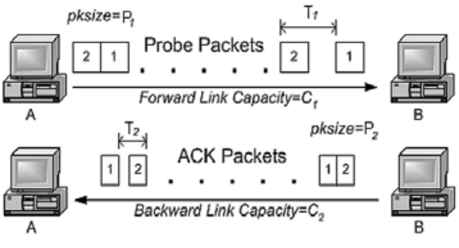


Fig. 2. Interaction of probe packets in asymmetric link scenarios

Table 1. Estimate asymmetric link capacity by varying packet sizes (ideal case without cross traffic and any queuing delays)

Probe (P_1) and ACK (P_2) Packet Size	Measured Dispersion T'	Capacity Estimation C'_1 and C'_2
$\frac{P_2}{C_2} > \frac{P_1}{C_1}$	$T' \rightarrow T_2$	$C'_1 < C_1; C'_2 = C_2$
$\frac{P_2}{C_2} < \frac{P_1}{C_1}$	$T' \rightarrow T_1$	$C'_1 = C_1; C'_2 < C_2$
$\frac{P_2}{C_2} = \frac{P_1}{C_1}$	$T' \rightarrow T_1 = T_2$	$C'_1 = C_1; C'_2 = C_2$
$P_2 = P_1$	$T' \rightarrow max(T_1, T_2)$	$C'_1 = C'_2 = min(C_1, C_2)$

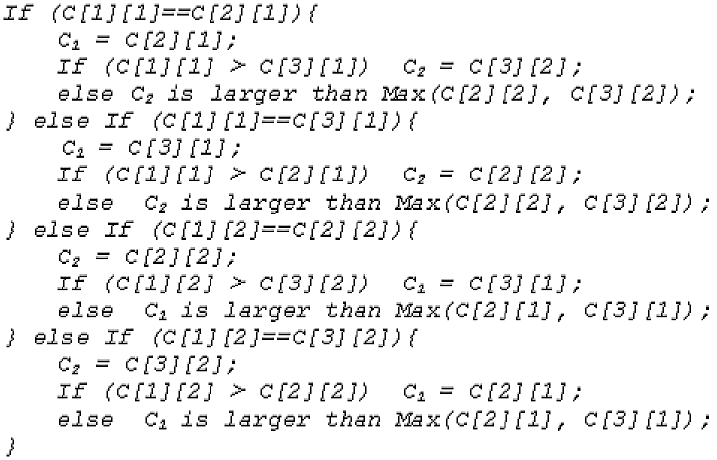
The dispersion measured at the end host A, denoted as T', is the dispersion between back-to-back acknowledgement packets. Suppose $T_1 > T_2$, this means that measured dispersion T' equates to T_1. We assume that host B immediately acknowledges the probe packets without incurring additional queuing delay, else the min sum condition would be violated and the pair discarded. On the other hand, suppose $T_1 < T_2$, then T' reflects T_2 instead, i.e. the dispersion generated on the backward direction link prevails. Therefore,

$$T' = max(T_1, T_2) \tag{2}$$

By varying the packet size ratio between the probe and the ACK packets, and observing the forward link capacity estimate (C'_1, which is $\frac{P_1}{T'}$) and the backward link capacity estimation (C'_2, which is $\frac{P_2}{T'}$), the source node can obtain the correct capacity estimations for both directions of the link. For instance, suppose $C_1 > C_2$ and the initial packet size $P_1 = P_2$, it can be concluded that $T_2 > T_1$ because $\frac{P_2}{C_2} > \frac{P_1}{C_1}$. From the discussion above, the end-to-end dispersion T' measured equates to T_2. Therefore, $C'_2 = \frac{P_2}{T'} = C_2$ and $C'_1 = \frac{P_1}{T'} = C_2 < C_1$. The estimated capacity is the round-trip bottleneck link capacity on the asymmetric link (the minimum value of C_1 and C_2), which is exactly is what CapProbe estimates as presented in [4].

However, by increasing P_1 gradually, C'_2 remains equivalent to C_2, but C'_1 increases and approaches C_1 gradually. When P_1 increased to $\frac{P_2 \times C_1}{C_2}$, C'_1 converges to C_1 and C'_2 converges to C_2. Conversely, after P_1 increased to larger than $\frac{P_2 \times C_1}{C_2}$ (i.e. $T_1 > T_2$, since $\frac{P_1}{C_1} > \frac{P_2}{C_2}$), T' will reflect T_1. As a result, $C'_1 = \frac{P_1}{T'} = C_1$ and $C'_2 = \frac{P_2}{T'} < C_2$. This simple relationship between the estimated capacity (C'_1, C'_2) and the varying packet size can be harvested for the accurate estimation of asymmetric link capacities. Table 1 below details this relationship.

Based on the relationship presented in the table, the AsymProbe algorithm consists of four phases, of which the first three phases are *Probing* phases, and the last is the *Decision* phase. Two packet sizes are used in the probing phases: P_{max} and P_{min}, which are chosen carefully by taking network and system issues into account. In the first probing phase, P_1 and P_2 are both set to P_{max}. Thus we estimate the bottleneck capacity, C_{low}, of the round trip path. In phase 2 and 3, (P_1,P_2) are set first to (P_{max},P_{min}) and then to (P_{min},P_{max}) in order to



```
If (C[1][1]==C[2][1]){
    $C_1$ = C[2][1];
    If (C[1][1] > C[3][1])  $C_2$ = C[3][2];
    else $C_2$ is larger than Max(C[2][2], C[3][2]);
} else If (C[1][1]==C[3][1]){
    $C_1$ = C[3][1];
    If (C[1][1] > C[2][1])  $C_2$ = C[2][2];
    else  $C_2$ is larger than Max(C[2][2], C[3][2]);
} else If (C[1][2]==C[2][2]){
    $C_2$ = C[2][2];
    If (C[1][2] > C[3][2])  $C_1$ = C[3][1];
    else  $C_1$ is larger than Max(C[2][1], C[3][1]);
} else If (C[1][2]==C[3][2]){
    $C_2$ = C[3][2];
    If (C[1][2] > C[2][2])  $C_1$ = C[2][1];
    else  $C_1$ is larger than Max(C[2][1], C[3][1]);
}
```

Fig. 3. AsymProbe Algorithm (The *Decision* Phase)

estimate the forward and backward link capacities respectively. We use $C[i][1]$ and $C[i][2]$ to denote the estimation results of C_1' and C_2' in the i-th phase, respectively.

In the fourth phase, namely the *Decision* phase, a decision algorithm is performed to determine the estimation results of both direction links from all $C[i][1]$ and $C[i][2]$ as shown in Fig. 3. However, it should also be mentioned that the capability of AsymProbe in determining the larger capacity is mathematically bounded by the maximum ratio of packet sizes between probe and acknowledgement packets, i.e. the max of $\frac{P_1}{P_2}$ and $\frac{P_2}{P_1}$. Specifically, if the capacity estimates of all 3 phases are equal, and we know a priori that the link is asymmetric, then, the packet size ratio is not sufficient large to provide an accurate capacity estimation of the larger link. Therefore, AsymProbe is unable to estimate the actual capacity in the direction with higher speed, but will indicate that such condition has occurred and report a "lower-bound" (i.e. $\frac{P_{max}}{P_{min}} \times C_{low}$) of the capacity instead. In this case, one-way capacity estimation tools (e.g. one way version of CapProbe or Pathrate) can be applied to accurately measure the capacity in this direction - if this solution is feasible within the scope of the application. In section 5.4, we discuss another extension of AsymProbe to this problem.

4 Simulation

In this section, we present simulation results that evaluate the accuracy of capacity estimation of AsymProbe on paths with asymmetric links. AsymProbe is implemented in the NS-2 simulator [8]. Fig. 4 depicts the simulation topology that represents a commonly seen scenario nowadays with an asymmetric DSL link. All links are symmetric 100Mbps Ethernet links except the one between node D and E, which is an asymmetric DSL link with 1.5Mbps downlink capacity (from E to D) and 128Kbps uplink capacity (from D to E). Nodes to the

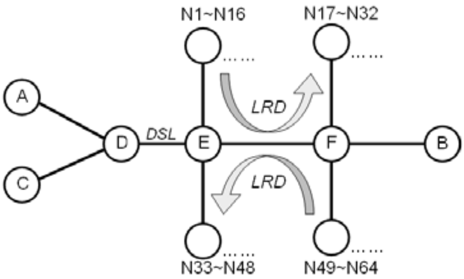


Fig. 4. Last-hop ADSL scenario. The link capacities are 100Mbps for all links, except the asymmetric DSL link between D and E ($D \rightarrow E : 128Kbps; E \rightarrow D : 1.5Mbps$)

Table 2. Simulation results of AsymProbe in last-hop DSL scenarios (Unit: Kbps)

Cross Traffic	AsymProbe from A		AsymProbe from B		CapProbe
	$A \rightarrow B$	$B \rightarrow A$	$B \rightarrow A$	$A \rightarrow B$	$A \Leftrightarrow B$
none	128	1500	1500	128	128
FTP ($B \rightarrow C$)	128	1500	1505	128.057	128
FTP ($C \rightarrow B$)	128.062	1500	1500	128	128
Poisson ($B \rightarrow C$, rate=300Kbps)	128	1500	1500	128	128
Poisson ($B \rightarrow C$, rate=750Kbps)	128	1500	1500	128	128
Poisson ($B \rightarrow C$, rate=1500Kbps)	128	1500	1500	128	128
Poisson ($C \rightarrow B$, rate=25.6Kbps)	128	1500	1500	128	128
Poisson ($C \rightarrow B$, rate=64Kbps)	128.006	1500	1500	127.936	128
Poisson ($C \rightarrow B$, rate=128Kbps)	127.988	1500	1483.143	128.039	128

left of node D (namely A and C) belong to a home networks, while nodes to the right of node E are on the Internet.

The AsymProbe estimation is performed on the path between node A and B. In addition to the AsymProbe flow, various types of cross traffic were generated on the DSL link to test AsymProbe robustness. The cross traffic types used were FTP and Poisson based UDP traffic of different rates. For the Internet segment, long range dependent (LRD) traffic is created between node E and F in both directions. The LRD traffic is composed of 16 Pareto flows with alpha = 1.9 [7], and the overall rate of LRD traffic is 60Mbps in each direction.

The maximum and minimum AsymProbe packet sizes, P_{max} and P_{min}, are set to 1500 bytes and 100 bytes respectively. For the various cross traffic configurations described in Table. 2, AsymProbe is independently initiated from both A and B; results obtained from AsymProbe are then compared against CapProbe as summarized in Table 2.

From the results shown in Table 2, AsymProbe is able to estimate the correct link capacity in both directions for all test cases; whereas CapProbe can only estimate the bottleneck link capacity of the round-trip path. Moreover, simulation results also show that AsymProbe works when placed on either the end-client

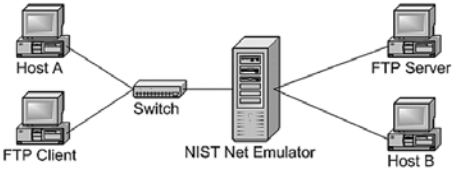


Fig. 5. Testbed for NIST Net experiments

(node A) or the Internet server (node B). The results are consistent in both cases.

It is also worth mentioning that since the link capacity ratio of the simulated scenario is 1.5Mbps/128Kbps, it is smaller than the packet size ratio $\frac{P_{max}}{P_{min}}$, AsymProbe is thus able to measure the correct link capacities. However, if we decrease the packet size ratio (e.g. increasing P_{min} in order to avoid the fine time resolution problem as described in [5]) and obtain a packet size ratio that is larger than the link capacity ratio, AsymProbe will only estimate the correct capacity of the narrower link and output the other direction link as a lower bound estimation, defined as $\frac{P_{max}}{P_{min}} \times C_{low}$. In such case, one-way link capacity estimation tools can be launched.

5 Experiments

In this section, we present testbed and Internet experimental results to further evaluate AsymProbe. We first perform a set of experiments on a "controlled" testbed to calibrate and verify the correctness of the AsymProbe scheme and its Linux implementation. We then move to Internet measurements for an evaluation in the diverse and realistic scenario.

5.1 Testbed Experiments

The testbed experiments are performed in the configuration shown in Fig. 5. The NISTNet emulator [9] is used to set up the asymmetric bottleneck link of various capacities. A backlogged file transfer session is generated from the FTP server to the client as cross traffic. This FTP connection shares the bottleneck link with an AsymProbe connection that traverses from host A to B.

For reasons we will discuss shortly, we choose 1500 bytes and 500 bytes as the maximum and minimum packet sizes in this set of experiments, respectively. Thus we have $\frac{P_{max}}{P_{min}} = \frac{1500}{500} = 3$. We present the experiment results in Table 3 and Table 4, in which we may see that when the forward/backward (Table 3) or backward/forward (Table 4) capacity ratio is below 3, AsymProbe measures both forward and backward capacities very accurately. When the ratio increases beyond 3, only a lower bound can be obtained in the direction with the larger capacity.

Table 3. NIST Net results on High/ Low asymmetric links (Unit: Mbps)

Link Capacity		AsymProbe Estimation		CapProbe Estimation
F	B	F	B	
1	1	1.063	1.064	0.981
		1.064	1.065	0.981
		1.063	1.063	0.985
2	1	2.010	1.063	0.979
		2.015	1.062	0.979
		2.010	1.064	0.979
3	1	2.997	1.065	0.985
		3.012	1.065	0.983
		3.015	1.055	0.981
4	1	≥3.611	1.064	0.979
		≥3.609	1.061	0.981
		≥3.611	1.062	0.979

F: Forward Link; B: Backward Link

Table 4. NIST Net results on Low/ High asymmetric links (Unit: Mbps)

Link Capacity		AsymProbe Estimation		CapProbe Estimation
F	B	F	B	
1	1	1.063	1.065	0.981
		1.065	1.064	0.981
		1.065	1.065	0.979
1	2	1.064	2.015	0.979
		1.062	2.010	0.975
		1.006	1.989	0.981
1	3	1.062	2.997	0.983
		1.063	3.018	0.985
		1.056	2.997	0.981
1	4	1.059	≥3.610	0.981
		1.065	≥3.610	0.979
		1.065	≥3.611	0.979

F: Forward Link; B: Backward Link

5.2 Internet Experiments

In addition to the controlled testbed experiments, we also perform a set of Internet measurements to evaluate AsymProbe in a more diverse and realistic scenario. In this set of experiments, again we have $\frac{P_{max}}{P_{min}} = \frac{1500}{500} = 3$. However, the asymmetric links we have found, provided by DSL [1] and Cable [2] companies, all have a higher down-link/up-link capacity ratio than 3. As presented in Table 5, AsymProbe captures the up-link capacities accurately, while only obtaining lower bounds for the down-links.

5.3 Discussion

From the simulation and experiment results above, AsymProbe is capable of estimating asymmetric link capacities, as long as the capacity ratio of the forward and backward links is within the range of the packet size ratio of the employed probe and acknowledgement packets. In order to increase the estimation range of AsymProbe, the packet size ratio should be as large as possible. However, this ratio is bound by implementation.

Specifically, the maximum size of the employed packets must be bounded by the Maximum Transmission Unit (MTU), which is the largest size of an IP datagram allowed to transmit on the path without fragmentation. The size of MTU may vary greatly in different system configurations. However, practically it is set to 1500 bytes in most networks. Packets larger than MTU will be seg-

[1] DSL 1 is provided by Verizon: *http://www.verizon.com*; DSL 2 is provided by Hinet: *http://www.hinet.net*

[2] Cable Modem is provided by Comcast: *http://www.comcast.com*

Table 5. Internet results on asymmetric link

Link	Claimed Capacity		Estimated Capacity		
	Down	Up	#	Down	Up
DSL 1	1.5 Mbps	128 Kbps	1	$\geq$ 379 Kbps	132 Kbps
			2	$\geq$ 382 Kbps	132 Kbps
			3	$\geq$ 380 Kbps	132 Kbps
DSL 2	3 Mbps	512 Kbps	1	$\geq$ 1.49 Mbps	565 Kbps
			2	$\geq$ 1.53 Mbps	567 Kbps
			3	$\geq$ 1.51 Mbps	558 Kbps
Cable Modem	3 Mbps	256 Kbps	1	$\geq$ 721 Kbps	247 Kbps
			2	$\geq$ 730 Kbps	255 Kbps
			3	$\geq$ 723 Kbps	248 Kbps

mented into smaller fragments for transmission and then reassembled on the receiving host. Therefore, using packets larger than MTU is not appropriate for CapProbe-based capacity estimation techniques, since the dispersion measurement no longer reflects the bottleneck capacity.

On the other hand, the minimum size of AsymProbe packets is also bounded in accordance with the supported time resolution on the estimating host. This is due to the fact that a packet pair with a smaller packet size will result in a smaller inter-packet dispersion, which in turn requires a finer time resolution to be measured accurately. Assume the capacity of the narrow link is C and the probing packet size is P, the dispersion time (and also the clock granularity needed for accurate estimation) that needs to be measured is $T = P/C$. Table 6 shows the required clock granularities that are needed for different probing packet sizes and narrow link capacities.

Table 6. Required time resolution for accurate estimation

Packet Size	Narrow Link Capacity			
	1 Gbps	100 Mbps	10 Mbps	1 Mbps
100 bytes	0.0008 ms	0.008 ms	0.08 ms	0.8 ms
500 bytes	0.004 ms	0.04 ms	0.4 ms	4 ms
1000 bytes	0.008 ms	0.08 ms	0.8 ms	8 ms
1500 bytes	0.012 ms	0.12 ms	1.2 ms	12 ms

It is clear that the time resolution of an end host relies on the hardware speed and the operating system. A system with fast processors and I/O interfaces can provide a finer time resolution. Additionally, [5] also shows that the accuracy of CapProbe-based capacity estimation is tightly related to the runtime execution mode. With kernel mode implementations, the capacity estimation is faster and more accurate than with user mode implementations. Kernel mode implementations also provide better time resolutions. Therefore, kernel mode implementations can use smaller packets for capacity estimation than the user mode implementations.

In the presented testbed experiments, the employed packet sizes are bounded with 1500 bytes as the maximum and 500 bytes as the minimum. The value of 1500 bytes is determined by the MTU on the path, whereas the value of 500 bytes is the minimum packet size which can measure the dispersion accurately with the provided machine time resolution. Thus it is only capable of estimating an asymmetric link with capacity ratio up to 1500 : 500, namely 3 : 1. For those links with even higher "asymmetric ratios" (e.g. 1.5Mbps/128Kbps DSL links or 400Kbps/64Kbps satellite links), it is necessary to increase the packet size ratio by either increasing the maximum packet size or decreasing the minimum packet size. In such cases, using a faster machine or switching from user mode to kernel mode can help.

If all of the above procedures do not work, one can resort to one way capacity estimation as mentioned in Section 3. This, however, requires full implementation on the receiver. If the receiver does not cooperate, a possible solution is to use a packet "train" probing concept as suggested by other researchers [1]. The intent is to replicate the effect of a "long" probing packet without paying the penalty of reassembly at the host. To illustrate the technique, consider for example the situation of an asymmetric satellite link with 15Mbps downlink and 128Kpbs uplink capacity. The server in the Internet must determine the downlink speed to deliver the proper content to a mobile user. The downlink capacity estimation can be achieved by transmitting a train of 10 consecutive 1500 byte packets, with a Probe leader and Probe trailer. As before, the two Probe packets each trigger a 100 byte packet probe response from the receiver. The train dispersion in the forward link is preserved in the ACK dispersion measured by the sender after the round trip and provides the desired estimate. Basically, this scheme is an extension of AsymProbe, where the source experiments with trains of increasing length until success. As pointed out in [1], the longer the train, the less dominant the mode corresponding to the forward narrow capacity. Consequently, the less frequent the train samples where no delay/interference occurred along the path and thus the less accurate the measurements. The measurement however, provides a conservative (lower bound) estimate of the narrow forward capacity, which can be progressively improved as more and more samples are collected. By the way, the "average" capacity measurement (as opposed to min sum measurement) was shown to converge for large train length N to a value between the narrow link and the residual link bandwidth. As expected, the lower the utilization, the faster the min sum measurement convergence [1].

6 Conclusions

In this paper, we studied asymmetric link capacity estimation and proposed an extension of CapProbe, namely AsymProbe, to estimate asymmetric link capacities. By strategically altering the ratio of probe and acknowledgement packet sizes, AsymProbe can simultaneously measure the link capacities of both forward and backward direction links. Through simulation and testbed experiments, we validated the accuracy and capabilities of our proposed approach.

The unique advantage of AsymProbe is the ability to measure capacities from the server using a round trip method that does not require the cooperation of the receiver (which may have limited processing power or may be altogether unaware of AsymProbe). Moreover, the technique is extremely fast, thus it is suitable for mobile receivers that experience rapidly varying, often asymmetric Internet connectivity. The simplicity, accuracy, and speed of AsymProbe make it ideal in real deployments where online and timely capacity estimation is required. Better service can be provided by estimating both forward/backward direction path capacities. Typical applications feature the efficient transfer of multimedia files over rapidly varying Internet paths (which may include wireless segments). Popular examples are P2P streaming and file sharing, overlay network structuring, and intelligent vertical handoff decision.

References

1. C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In Proc. of IEEE Infocom 2001.
2. A. B. Downey. Using Pathchar to Estimate Internet Link Characteristics. In Proc. of ACM SIGCOMM 1999.
3. V. Jacobson. Pathchar: A tool to infer characteristics of Internet paths. ftp://ftp.ee.lbl.gov/pathchar
4. R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, M. Y. Sanadidi. CapProbe: A Simple and Accurate Capacity Estimation Technique. In Proc. of ACM SIGCOMM 2004.
5. R. Kapoor, L.-J. Chen, M. Y. Sanadidi, M. Gerla. Accuracy of Link Capacity Estimates using Passive and Active Approaches with CapProbe. In Proc. of ISCC 2004.
6. K. Lai and M. Baker. Measuring Bandwidth. In Proc. of IEEE INFOCOM 1999.
7. M.S. Taqqu, W. Willinger, R. Sherman. Proof of a fundamental result in self-similar traffic modeling. SIGCOMM Computer Communications Review, 27: 5-23, 1997.
8. Network Simulator (NS-2). www.mash.cs.berkeley.edu/ns/
9. NIST Net. http://snad.ncsl.nist.gov/itg/nistnet/

Impact of Resource Sharability on Dual Failure Restorability in Optical Mesh Networks

Chadi Assi[1], Wei Huo[1], and Abdallah Shami[2]

[1] Concordia Institute for Information Systems Engineering,
Concordia University, Montreal, Quebec, Canada
assi@ciise.concordia.ca

[2] Department of Electrical and Computer Engineering,
The University of Western Ontario, London, Ontario, Canada
ashami@eng.uwo.ca

Abstract. Protection capacity re-provisioning mitigates the impact of double-link failures by provisioning new capacity for unprotected demands in a network designed to achieve 100% restorability under single-link failures. We study the performance of re-provisioning in networks under various resource sharing degrees. Intuitively, the lower is the sharability degree of resources the smaller is the number of unprotected connections resulting after the recovery from a failure. However, we show in this paper that limited resource sharability also implies limited flexibility for the network in finding capacity for unprotected demands after failures; which accordingly limits the capability of re-provisioning schemes in improving the network restorability. We further study the performance of different re-provisioning algorithms under distributed control and we show that contentions severely impact the restorability performance. Finally, we propose a simple mechanism to mitigate the effects of contentions.

1 Introduction

Significant progress has been made towards making optical networks resilient in the event of single link failures. Protection schemes with preplanned spare capacity [1] have been extensively studied in optical mesh networks, where protection capacity can either be dedicated or shared among multiple connections whose primary paths are physically disjoint [2, 3, 4]. Now as the size and the complexity of optical mesh networks continue to grow, dual failures become increasingly probable [6, 7]. Hence designing recovery algorithms to protect against such failure events and ensure service continuity is a paramount concern.

To date, various research efforts have already addressed the problem of routing connections under dual failure assumptions, and findings show that designs offering complete dual-failure restorability require more than double the amount of spare capacity [5, 14]. In order to avoid this excessive deployment of extra spare capacity in the network, *capacity reconfiguration* after the occurrence of and recovery from the first failure has been proposed [6-10]. After the occurrence

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 792–803, 2005.

of the first failure, the failed connections are restored from their working paths into their protection paths. Hence upon complete recovery, backup capacity along now active protection routes can no longer be shared. As a result some of the connections in the network will become indirectly unprotected, therefore increasing the network vulnerability to a subsequent failure. Capacity re-provisioning provides a mechanism by which one can find and allocate new protection capacities for these newly unprotected connections without a priori knowledge of the location of the second failure. Many backup re-provisioning algorithms for handling multiple failures have recently been proposed [6, 7, 8, 10, 14] and comprehensive studies indicate that re-provisioning can dramatically lower network vulnerability.

In this paper we discuss a new re-provisioning scheme to improve the connection restorability in optical shared mesh networks. We assume two near-simultaneous failures, where the second failure occurs after the first failure is recovered from, but before it is physically repaired. A critical objective for re-provisioning is to reduce the total number of connections that have to be re-provisioned. Here the motivations are twofold: (1) to reduce management overheads in simultaneously provisioning a large number of connections, and (2) to lower reservation contention between multiple unprotected connections trying to establish backup capacity. The latter may result in increased blocking rates for re-provisioning, which in turn will increase vulnerability to subsequent failure(s).

To better utilize the network resources, connections may be allowed to share their protection wavelengths if their corresponding working paths are link disjoint. These protection resources are activated upon the occurance of a failure to restore the affected connections and therefore cannot be shared anymore, leaving some connections unprotected in the network. The number of resulting unprotected connections can be very large if the degree of sharability (also referred to as sharability index) of protection wavelengths is high. Accordingly, limiting the resource sharability in the network will result in reducing the number of unprotected demands. However, since re-provisioning makes use of available resources in the network to provision new protection capacity, limited sharability will yield lower flexibility is finding and assigning resources. Therefore, it is clear that there are two conflicting design constraints: on one hand limited SI may reduce the number of unprotected connections but at the expense of less flexibility in allocating protection capacity for unprotected connections; on the other hand, higher SI may result in larger number of unprotected connections after the first failure with higher degree of flexibility in provisioning protection capacity. Hence, one objective of the paper is to provide a comprehensive study on the performance of capacity re-provisioning under different sharability degrees.

Another factor that may limit the performance of re-provisioning is the underlying implementation of the protocol, namely centralized and distributed control. In this paper, we compare the performances of re-provisioning under the two implemenations. Furthermore, we propose a simple and efficient technique to cope with the adverse effects of contentions incurred in distributed re-provisioning. The rest of the paper is organized as follows. In section 2 we study the impact of resource sharability on network restorability. In section 3 we introduce a new re-provisioning algorithm and in section 4 we discuss the performance under centralized and distributed control. Section 5 presents performance evaluation and comparisons and finally we conclude in section 6.

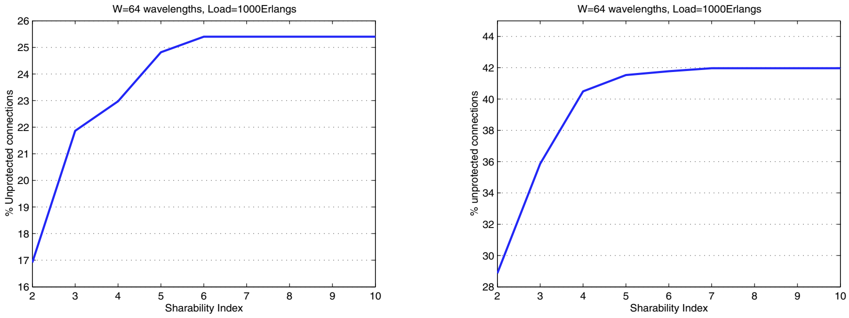
2 Impact of Sharability on Network Restorability

Under normal conditions, the network is usually protected against all single failures. Namely, when a failure occurs, all connections whose working paths are affected by that failure are re-routed on their corresponding protection paths [2, 3, 4]. Hence, a connection recovery usually requires source node notification and recovery signaling to configure the protection resources (e.g., wavelengths and cross connect switches) along the backup route [4]. However, since these protection resources may also be shared with other unaffected connections, these may become unprotected and vulnerable to the next failure [10]. Overall, to summarize these unprotected connections, one can classify them into three categories:

1) *Indirectly Affected Connections*: Upon failure, shared protection resources are activated by the failed connections which may cause some connections (whose backup lightpaths share these protection resources) to become unprotected.
2) *Directly Affected Working Connections*: A failed demand that is re-routed to its backup path is still vulnerable to a second failure that may affect its active protection path.
3) *Directly Affected Backup Connections*: Demands whose protection connections have failed due to the first failure.

Clearly, increased numbers of unprotected connections in the network can increase its vulnerability to subsequent failures and hence lower the overall network restorability. To improve the service availability, re-provisioning exploits the available capacity in the network to re-establish new backup paths for unprotected connections in advance of a failure (and right after the recovery from the first fault). One such scheme has been proposed in [8, 10] and its performance is evaluated here as well, termed thereafter as Scheme I. Namely, after recovery from the first failure, the algorithm categorizes all unprotected connections into one of the three categories detailed above. Subsequently, it attempts to establish new protection lightpaths for these demands.

Note that the degree of wavelength sharability (i.e., the number of connections allowed to share the same protection resources) directly impacts the number of unprotected connections that needs re-provisioning. Clearly, if this sharability degree is large, the network will potentially admit more connections since more backup paths are packed together; and a large number of connections will become vulnerable after the first failure (i.e., the impact of connections in category 2 to those in category 1). Conversely, limiting the sharability of a protection wavelength will reduce the total number of unprotected connections after the first failure; however that comes at the expense of reduced network performance that limited sharability yields.



(a) No wavelength conversion

(b) Full wavelength conversion

Fig. 1. Percentage of Unprotected Connections before Re-provisioning vs. SI

Fig.1 shows a case on a nation wide network (Fig. 3) where the percentage of unprotected connections in measured after the recovery from the first failure by varying the sharability index, SI. Clearly, the figure shows a substantial increase in the number of unprotected connections (e.g., 9% increase in wavelength continuous network and 13% in a wavelength convertible network [15]) in the network after the first failure if unlimited resource sharability is allowed. Therefore, intuitively this finding suggests that limiting the sharability index will result in a lower number of unprotected demands after recovery and therefore in a less number of connections to be re-provisioned. However, on the other hand, a smaller number of unprotected demands to be re-provisioned does not always guarantee that the percentage (or the number) of unprotected connections in the network after re-provisioning is reduced. The reason for that is that limited resource sharability will effectively limit the performance of the re-provisioning algorithm in finding protection resources for unprotected demands. Moreover, limited sharability will limit the overall performance of the network since fewer connections can be accommodated by the network. Alternatively, unlimited resource sharability may provide the necessary flexibility in allocating protection capacity to unprotected demands after the first failure but, as the figure shows a larger group of demands become unprotected and require re-provisioning. Moreover, if distributed re-provisioning is implemented, larger number of connections attempting to reserve protection capacity will amplify the contentions over resources, yielding higher blocking and leaving more connections unprotected even when resources are available. Therefore, it is clear that there are two conflicting design constraints: on one hand limited SI may reduce the number of unprotected connections but at the expense of limited network performance and less flexibility in allocating protection capacity for unprotected connections. On the other hand, higher SI may yield a larger number of unprotected connections after the first failure and magnifies the effect of contentions under distributed control but yields higher degree of flexibility in provisioning protection capacity.

3 Network Re-provisioning

3.1 An Alternative Approach

A critical objective for re-provisioning is the reduction of the number of unprotected connections after the first failure. Achieving this objective strongly depends on the resource availability in the network after the failure and the limit imposed on their sharability. Here, when a failed connection is recovered onto its backup lightpath, a large number of unaffected demands, whose backup connections share the same protection capacity on any link along the backup of the failed connection, will be unprotected and hence require re-provisioning. Typically, the larger the sharability of a wavelength, the larger will be the number of indirectly affected connections in the network that become unprotected. Therefore, if only a new working lightpath is setup for each failed connection, then traffic is switched back from the backup lightpath to the new working path and the shared protection capacity becomes available leaving the indirectly affected connections protected; we refer to this methodology thereafter as Scheme II [16].

The effectiveness of Scheme II is best shown via an illustrative example in Fig.2. We assume initially b_1, b_2 and b_3 are all setup using λ_1, and b_1 shares λ_1 on link (D-E) with b_2 and on link (E-H) with b_3. When link (B-F) fails, w_1 is restored to its backup b_1 and as a result, b_2 and b_3 become unavailable since they share protection capacity with b_1. Hence b_1, w_2 and w_3 become all unprotected and three new protection paths (or capacity) need to be re-provisioned in order to fully protect the network against a subsequent failure. Under Scheme II however, when w_1 is restored to its backup, connection b_1, w_2 and w_3 become *temporarily unprotected*. Hence, if we can find a new working path (w_1^{new}) that is link disjoint with b_1 to carry the failed traffic, then b_2 and b_3 can also become available again and their corresponding connections (w_2, w_3) are fully protected. Note that w_1^{new} may not be disjoint with w_2 and/or w_3 (w_2 in this example). Therefore, b_1 cannot share any protection resource with b_2. In a wavelength continuous network, a new backup b_1^{new} (and protection wavelength) that is link-disjoint with w_1^{new} has to be provisioned. In a wavelength convertible network, the conflict links are identified (e.g., (D-E)) and a different wavelength is provisioned along those links (e.g., λ_2 can be assigned to b_1 on link (D-E) leaving the rest of the backup lightpath intact). Note that Scheme II differs from Scheme I in that the number of connections to be re-provisioned upon a failure is dramatically reduced, whereas the number of *temporarily unprotected* connections during the re-provisioning time remains the same. Finally, Scheme II has a strong impact on the overall network restorability. Even under higher SI, the number of connections to be re-provisioned can be much reduced. The number of connections to be re-provisioned under Scheme II is substantially lowered resulting in a less management overhead and lighter impacts of contentions under distributed control.

Now clearly, under the premise that only new working path for failed connections are provisioned, resource sharability may have minimal impact on the

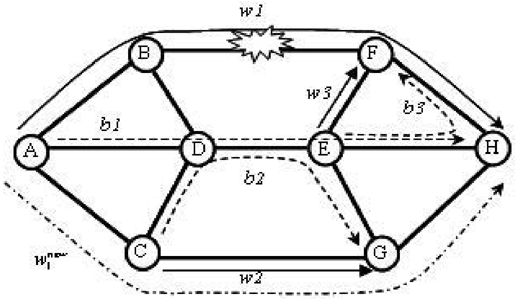


Fig. 2. Example for Re-provisioning

performance of the re-provisioning algorithm since working capacity can not be shared. Notice, however, that Scheme II does not neglect completely the effect of resource sharability; when a link fails, protection wavelengths on this link will also fail and these wavelengths may be protecting a larger number of connections when the SI is higher. Moreover, if the algorithm does not succeed in provisioning (some) new working connections for failed demands, then (as in Scheme I) the new scheme will identify the unprotected connections (those in categories 1 and 2) and re-provision them. Accordingly, the SI will have an impact on increasing the number of connections to be re-provisioned.

3.2 Implementation Perspective

We have seen that the performance of re-provisioning strongly depends on the algorithm itself and the sharability of protection resources. Another factor that also affects the performance is the implementation of the underlying algorithm; an algorithm typically can either have a centralized or a distributed control [11].

Under a centralized implementation, a central network management system holds the global information of network resources, such as network topology, link states, wavelength usage on each link, sharability information for protection resources, etc., and the corresponding steps of the particular algorithm are executed at this central controller. Here, upon the occurrence of a failure, the network will take the responsibility of recovering the failed connections through a standard signaling recovery protocol [4] and the central controller is informed through an alarm message to initiate the re-provisioning procedure of each unprotected demand. Clearly, centralized re-provisioning of unprotected connections is done sequentially in order to avoid contention for capacity. Contentions for capacity may lead to increasing the number of unprotected connections in the network and therefore increasing the vulnerability to a subsequent failure.

Alternatively, under distributed implementation, the source node of each unprotected demand is responsible for re-provisioning new protection capacity for its connection. An unprotected demand is typically identified by either the node detecting the link failure or by the source node of a failed connection. We deploy here a distributed provisioning approach with forward reservation [12], whereby the source node of one unprotected demand computes a new path and/or a new protection wavelength. Subsequently the node sends a control message containing the new selected wavelength to reserve resources along the entire path. If at least one node along the route is not successful in reserving the selected wavelength, the reservation fails and the connection is deemed unprotected. Here, unlike the centralized scheme where all connections are re-provisioned sequentially, all unprotected connections attempt to reserve protection capacity simultaneously and therefore contentions [11, 12, 13] may likely occur among connections requesting the reservation of the same resource. Clearly, a connection failing to successfully find new protection capacity will be left unprotected and will ultimately increase the network vulnerability to a subsequent failure.

Note that, if the number of unprotected connections resulting from the first failure and simultaneously attempting to re-provision new protection capacity is quite large, contentions over resources is more likely to increase; thereby, leaving a large number of unprotected demands in the network upon re-provisioning. Hence, a larger sharability index will yield a larger group of unprotected connections and accordingly the intensity of contentions under distributed re-provisioning will be magnified. Therefore, to achieve better network restorability, the effect of contentions will have to be reduced. Here, one advantage Scheme II possesses over Scheme I is that the number of connections to be re-provisioned is potentially much smaller; therefore making the impact of contentions on network restorability less severe. Nonetheless, it is still a concern as it will prevalent in the next section. To mitigate the impacts contentions may have on the network restorability, we propose that unprotected connections attempting to re-provision and failing to succeed due to contention, be allowed to reattempt after selecting a different wavelength if possible. The advantage of reattempting is that blocking due to contentions may be reduced whereas the drawbacks are increased network re-provisioning times. Later we will see that re-provisioning retries strongly reduce the impact of contentions, and accordingly improves the dual failure network restorability.

4 Simulation Results

In this section, we present some numerical examples to illustrate the performance of capacity re-provisioning on improving the double failure restorability of optical networks and we study the impact that resource sharability may have on the performance. We implement a discrete event simulation tool using C++ to simulation the re-provisioning algorithm. The sample network topology we used is shown in Fig.3. and it consists of 24 nodes and 86 unidirectional links. Requests are uniformly distributed between all source-destination pairs and arrive at each node via a Poisson process with a mean arrival rate of λ arrivals/ms. The connection-holding time is exponential distributed and the number of wavelengths (W) per link is 64.

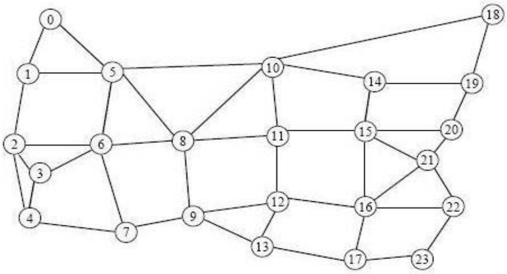
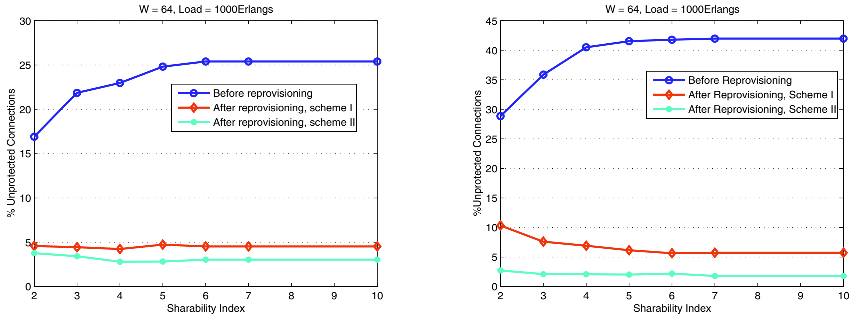


Fig. 3. A Sample Network Topology

We simulate the failure of a unidirectional link and we measure the percentage of the unprotected connections in the network after and before re-provisioning. Fig.4. shows the performance of the two re-provisioning algorithms, namely Scheme I and Scheme II, in a wavelength continuous (Fig.4a) and convertible (Fig.4b) network. As the sharability of protection resources increases, the figure shows that the percentage of unprotected connections in the network before re-provisioning increases clearly because more connections are admitted to the network. The SI here varies between 2 and 10 where a SI = 1 corresponds to the dedicated protection case. Clearly, capacity re-provisioning (under both schemes) improves the network performance by substantially reducing the percentage of unprotected connections (e.g., a decrease from 42% to 6% at higher SI in a wavelength convertible network using Scheme I) and therefore making the network less vulnerable to subsequent failures.

One interesting finding, shown in Fig.4a, is that of the impact of SI on the re-provisioning gain. Namely, the lower percentage of unprotected connections before re-provisioning at lower SI does not necessarily mean a good re-provisioning performance (i.e., a lower percentage of unprotected connections) after re-provisioning. The figure shows that as the SI increases, the percentage of unprotected connections after re-provisioning decreases for Scheme I (10%-6%) while it remains almost constant for Scheme II (∼3%) with better performance than that of Scheme I. The reason for this is that a lower SI will limit the flexibility of the re-provisioning algorithm in finding and judiciously allocating protection resources. While on the other hand, a higher SI will allow the network to accommodate more unprotected demands during re-provisioning. Therefore, the figure shows a larger performance gain at higher SI (38% (42%-6%)) than at lower SI (20% (30%-10%)). Alternatively, Scheme II shows a fixed percentage of unprotected connections at different sharability indices. The reason for that is due to the fact the Scheme II gives preference to provisioning new working capacity for failed demands in order to avoid re-provisioning a larger number of protection connections. Similar results are shown in Fig.4b, except that the percentage of unprotected connections is smaller since in a wavelength continuous network, fewer connections are admitted to the network due to the wavelength continuity constraint.



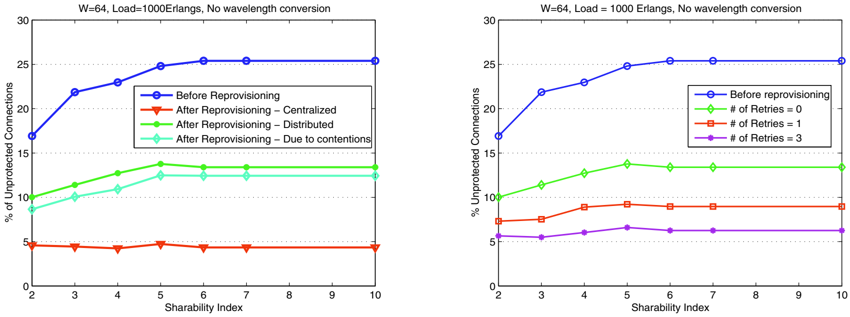
(a) No wavelength conversion

(b) Full wavelength conversion

Fig. 4. Percentage of Unprotected Connections - before and after Re-provisioning

Next, we present the performance of capacity re-provisioning under distributed control. As we have already mentioned, when a large number of unprotected connections attempt to re-provision simultaneously, contentions among resources will occur in the network. This will prevent some demands from protecting their connections although resources may be available. Fig.5a shows a comparison of re-provisioning for centralized and distributed implementation of Scheme I in a wavelength continuous network under different sharability degrees. Clearly, the percentage of unprotected connections after distributed re-provisioning is much higher than that under centralized re-provisioning (10%∼14% and 5%) which is mainly due to the effects of contentions. Moreover, at higher SI, by comparison to the performance under lower SI, a larger number of connections starts simultaneously the re-provisioning process and contends for the resources; hence stronger contentions may occur and as the figure shows, the percentage of unprotected connections increases. We should note that under distributed re-provisioning, a demand may fail to protect its connection for two reasons: (1) due to unavailable resources or (2) due to contentions with other connections. We measured the impact of contentions on increasing the number of unprotected connections and the results are shown in Fig.5a. Clearly, most of the connections fail to be protected due to contentions while attempting to reserve capacity. Here, unlike centralized scheme, in distributed re-provisioning the blocking due to insufficient resources decreases while the blocking due to contentions increases substantially due to the latency in receiving resource updates in time. Therefore, a node may attempt to reserve capacity that may already have been reserved by some other connection, leading to blocking due to contentions.

To minimize the impact of contentions, we propose that a connection that is being blocked due to only contention be allowed to select a new wavelength and retry its reservation. Here, when a demand fails to allocate protection capacity (wavelength) to its connection due to contentions, it will be allowed to reselect a different wavelength and re-attempt re-provisioning. Fig.5b shows the improve-



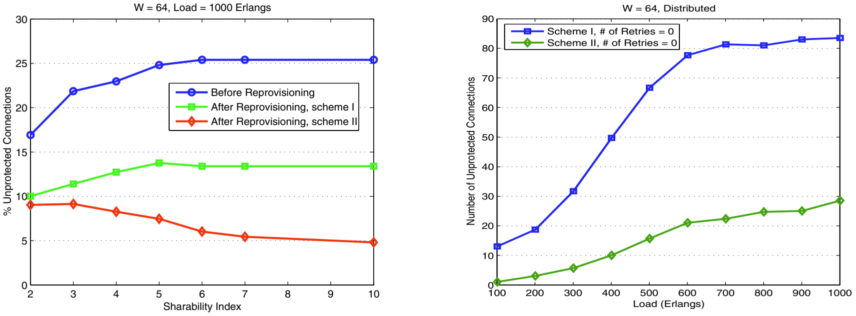
(a) Centralized vs. Distributed

(b) Distributed with Retries

Fig. 5. Effects of Contentions on the Performance of Re-provisioning, Scheme I

ment of this *re-select and reserve* scheme in reducing the percentage of unprotected connections after re-provisioning. The disadvantage of retrying, however, is the increase in the overall re-provisioning time. Our simulations showed that the total network re-provisioning time is kept below 1 second when the total number of retries is 3.

Finally, a comparison between Scheme I and Scheme II under distributed re-provisioning is presented in Fig.6a. Under Scheme I, the algorithm identifies all resulting unprotected connections after the recovery of failed connections (i.e., all connections in categories 1, 2 and 3) and those are all re-provisioned simultaneously. However, under Scheme II, the algorithm starts by re-provisioning only a fraction of those connections and namely, new working connections for the failed demands (i.e., those in categories 2 and 3); when unsuccessful, some of the connections in category 1 are identified and re-provisioned accordingly. Moreover, that fraction of demands is not re-provisioned simultaneously, therefore leading to lesser impact of contentions. Now when the SI is low, the performance of both schemes is similar (because fewer connections are packed together, hence the number of unprotected connections in category 1 is small) with Scheme II slightly outperforming Scheme I since the effect of contentions is not as strong. When the SI increases, the number of unprotected connections increases and accordingly contentions start to have larger impact on Scheme I; hence the increase in the percentage of unprotected connections after re-provisioning. However, under Scheme II the behavior is quite different. As SI increases, the percentage of unprotected connections after re-provisioning decreases (note that this does not necessarily mean that the total number of unprotected connections is reduced, since more unprotected connections may exist before re-provisioning) since the impact of contentions is not as severe and the algorithm judiciously uses the available capacity to protect more connections. As a conclusion, the effects of contentions on Scheme II is much smaller than that on Scheme I. Fig.6b shows the number of unprotected connections after re-provisioning that fail to reserve



(a) Percentage of Unprotected Connections

(b) Number of Unprotected Connections due to Contention

Fig. 6. Performance of Distributed Re-provisioning, Scheme I vs, Scheme II

protection capacity due to only contentions when the sharability of resources is very large. Clearly, the figure shows the strong impact contentions have on degrading the performance of Scheme I.

5 Conclusion

In this paper we studied the problem of improving the network restorability through protection capacity re-provisioning for unprotected connections after a failure. We showed that resource sharability plays a significant role in the performance of network re-provisioning. Namely, we showed that although limited resource sharability may yield to a small number of unprotected connections after a failure, it affects the performance of re-provisioning due to the limited flexibility in allocating protection capacity among unprotected demands. We studied the performance of re-provisioning under distributed control and we showed the strong impacts contentions may have on degrading the performance under different sharability conditions. We proposed a simple mechanism whereby contentions effects can be mitigated to improve the overall network restorability. We also presented a comparison of different re-provisioning schemes and we showed how the algorithm itself may strongly impact the overall performance.

References

1. W. Grover: "Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking", *prentice hall*, 2003.
2. S. Ramamurthy, B. Mukherjee: "Survivable WDM Mesh Networks, Part II - Restoration", *IEEE ICC 1999*, 1999.
3. J. Labourdette: "Shared Mesh Restoration in Optical Networks", *Proc. OFC 2004*, Feb. 2004.

4. J. Yates, G. Li: "Challenges in Intelligent Transport Network Restoration", *Proc. OFC 2003*, March 2003.
5. M. Clouqueur, W. D. Grover: "Mesh-restorable networks with complete dual failure restorability and with selectively enhanced dual-failure restorability properties", *SPIE OPTICOMM*, Boston, MA, July-Aug 2002.
6. D. Schupke, R. Prinz: "Performance of Path Protection and Rerouting for WDM Networks Subject to Dual Failures", *Proc. OFC 2003*, March 2003.
7. S. Kim, S. Lumetta: "Evaluation of Protection Reconfiguration for Multiple Failures in WDM Mesh Networks", *Proc. OFC 2003*, March 2003.
8. R. Ramamurthy, A. Akyamac, J-F. Labourdette, S. Chaudhuri: "Pre-Emptive Reprovisioning in Mesh Optical Networks", *Proc. OFC 2003*, March 2003.
9. P. Charalambous, et.al. "A National Mesh Network Using Optical Cross-Connect Switches", *Proc. OFC 2003*, March 2003.
10. J. Zhang, K. Zhu, B. Mukherjee: "A Comprehensive Study on Backup Reprovisioning to Remedy the Effect of Multiple-Link Failures in WDM Mesh Networks", *ICC04*, Paris, June 2004.
11. L. Shen and B. Ramamurthy: "Centralized vs. Distributed Connection Management Schemes under Different Traffic Patterns in Wavelength-Convertible Optical Networks", *Proc. IEEE ICC02*, NY, 2002.
12. Y. Mei, and C. Qiao: "Efficient Distributed Control Protocols for WDM Optical Networks", *Proc. ICCCN*, September 1997.
13. H. Zang, J. Jue, and B. Mukherjee: "A Review of Routing and Wavelength Assignment Approaches for Wavelength Routed Optical WDM Networks", *Optical Networks*, No.1, January 2000.
14. M. Clouqueur, W.D. Grover: "Availability Analysis of Span-restorable Mesh Networks", *IEEE JSAC*, vol.20, no. 4, May 2002, pp. 810-821.
15. B. Ramamurthy and B. Mukherjee: "Wavelength conversion in WDM networking", *IEEE J. Select. Area. Commun.*, vol. 16, no. 7, pp. 1061-1073, September 1998.
16. C. Assi, W. Huo, A. Shami, and N. Ghani: "Analysis of Capacity Re-Provisioning in Optical Mesh Networks", to appear in *IEEE Communications Letters*, (accepted in January 2005), 2005.

Efficient Distributed Solution for MPLS Fast Reroute

Dongmei Wang and Guangzhi Li

AT&T Labs- Research,
180 Park Avenue, Florham Park, NJ 07932
{mei,gli}@research.att.com

Ahstract. As service providers move more applications to their IP/MPLS (Multiple Protocol Label Switching) networks, rapid restoration upon failure becomes more and more crucial. Recently MPLS fast reroute has attracted lots of attention as it was designed to meet the needs of real-time applications, such as voice over IP. MPLS fast reroute achieves rapid restoration by computing and signaling backup label switched paths (LSP) in advance and re-directing traffic as close to failure point as possible. To provide a guarantee of failure restoration, extra bandwidth has to be reserved on backup LSPs. To improve the bandwidth utilization, path-merging technique was proposed to allow bandwidth sharing on common links among a service LSP and its backup LSPs. However, the sharing is very limited. In this paper, we provide efficient distributed solution, which would allow much broader bandwidth sharing among any backup LSPs from different service LSPs. We also propose an efficient algorithm for backup path selection to further increase the bandwidth sharing. The associated signaling extension for additional information distribution and collection is provided. To evaluate our solution, we compare its performance with the MPLS fast reroute proposal in IETF via simulation. The key figure-of-merit for restoration capacity efficiency is restoration overbuild, i.e., the ratio of restoration capacity to service capacity. Our simulation results show that our distributed solution reduces restoration overbuild from 2.5 to 1, and our optimized backup path selection further reduces restoration overbuild to about 0.5.

1 Introduction

As service providers move more applications to their IP/MPLS (Multiple Protocol Label Switching [1]) backbone networks, such as voice over IP and online games, et. al., rapid restoration upon failure becomes more and more crucial. Recently MPLS fast reroute has attracted lots of attention as it enables the re-direction of traffic onto backup LSPs (label switched path) within 50 milliseconds in the event of a failure [2]. Today, Internet Engineering Task Force (IETF) is in the process of standardizing Internet draft [2]. The Internet draft [2] addresses the necessary signaling extensions for supporting MPLS fast reroute, and proposes using path merging technique to share the bandwidth on common links among a service LSP and its backup LSPs. However, no solution is provided to solve the issues: how to share the bandwidth among any backup LSPs from different service LSPs, and how to select the backup LSPs to maximize the backup capacity sharing.

There has been a great deal of work addressing restoration functionality and schemes in IP/MPLS networks [3,4,5,6,11,12,13,14,20,22,24] as well as how to man-

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 804 – 815, 2005.

age restoration capacity and select optimized restoration paths without centralized server [7,8,9,10,23]. However, all of them deal with path-based end-to-end or segment-based restoration [21,22,24] except [13,14] addressing routing protocol fast re-convergence. To the best of our knowledge, we have not seen any published paper to address bandwidth sharing among any backup LSPs from different service LSPs, and how to select the backup LSPs to maximize the backup sharing for MPLS fast reroute. Since the backup LSPs are pre-established, even though they do not consume bandwidth until failure happens, the network has to reserve enough restoration bandwidth to guarantee the LSP restoration upon failure. Without bandwidth sharing among backup LSPs for different service LSPs, and without an efficient backup LSP selection algorithm, the network would reserve much more bandwidth on its links than necessary.

In this paper, we propose two schemes to achieve and maximize the bandwidth sharing among any backup LSPs from different service LSPs for MPLS fast reroute. The first scheme--an efficient distributed bandwidth management can be implemented with nearly zero effort since it does not require any routing/signaling extensions. With our proposed array operation at each node, the backup capacity sharing at each link is independent of the backup LSP selection algorithm. The second scheme--an efficient backup LSP selection algorithm requires extending signaling protocol to distribute and collect some additional routing information. But the signaling overhead is very small. As many other bandwidth sharing schemes [8,10,20,22,23], our solution is based on the observation that the restoration bandwidth can be shared by multiple backup LSPs so long as their protected service LSP path segments do not fail at the same time. Our proposals have no confliction with the IETF MPLS fast reroute draft [2], but rather enhancements to it.

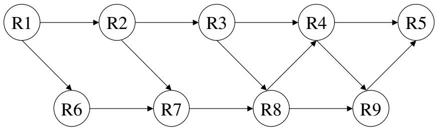
To demonstrate the efficiency of our proposals, we evaluate the bandwidth efficiencies of our approaches and the IETF MPLS fast reroute Internet draft [2] via simulation on two networks: a US intercity backbone network and a Toronto metropolitan network from [12], here we refer the Internet draft [2] as the baseline. The results show that our two enhancements reduce the amount of reserved restoration bandwidth or overbuild dramatically.

The paper is organized as follows. Section 2 summarizes MPLS fast reroute scheme and discusses its bandwidth management issues. Section 3 describes our first enhancement and gives an operational overview of our proposed bandwidth management method. Section 4 presents our second enhancement of efficient backup path selection algorithm and introduces new signaling messages between neighboring LSRs. Section 5 and section 6 present the simulation scenarios and performance results.

2 MPLS Fast Reroute

2.1 Baseline

IETF Internet draft [2] defines two MPLS fast reroute methods: one-to-one backup method and facility backup method. The one-to-one backup method creates detour LSPs for each protected service LSP at each potential point of failure, such as link failure or LSR failure. The facility backup method creates a bypass tunnel to protect a



Protected LSP:	R1->R2->R3->R4->R5
R1's backup:	R1->R6->R7->R8->R3
R2's backup:	R2->R7->R8->R4
R3's backup:	R3->R8->R9->R5
R4's backup:	R4->R9->R5

Fig. 1. MPLS fast reroute example

potential failure point. LSPs using the same protected facility will be protected via the same bypass tunnel. Thus, facility backup method can be pre-setup around potential failure point in the MPLS network and the bypass tunnels can be pre-designed in a centralized system to reduce network restoration capacity. However, the one-to-one backup method has to establish/delete backup LSPs on time as protected LSP comes and goes in a distributed MPLS network. In this paper, we only consider one-to-one backup method and propose mechanisms to maximize backup LSP capacity sharing on common links.

Figure 1 is an example from IETF Internet draft [2] illustrating one-to-one backup method. As [2] pointed out, to fully protect an LSP that traverses N nodes, there could be as many as N-1 backup paths. To minimize the number of LSPs in the network, it is desirable to merge a backup LSP to its service LSP when feasible. When a backup LSP intersects its service LSP at a LSR with the same outgoing interface, it will be merged. When two backup paths for the same protected service LSP travel a common link in the same direction, the required bandwidth can be shared since they are protecting different failure scenarios. When a failure occurs along the service LSP, the upstream LSR close to the failure point detects the failure and re-directs the traffic onto the local backup LSP. For example, if LSR R3 fails, R2 will detect the failure and redirect traffic along R2's backup path. Now the traffic will travel along R1-> R2->R7->R8->R4->R5. Since R2 cannot distinguish the failure between link R2->R3 and LSR R3, R2's backup usually excludes R3. Although backup LSPs do not consume bandwidth before failure, the network needs to reserve enough bandwidth along backup LSP links to guarantee 100% failure restoration. Using LSP merging technique, the bandwidth for service LSP and its backup LSPs can be shared along common links. The bandwidth for different backup LSPs of the same service LSP can also be shared along common links. But no solution is provided for backup LSP selection and bandwidth sharing among any backup LSPs from different service LSPs in [2] as we pointed out before. There is no surprising since Internet draft [2] is mainly focusing on MPLS fast reroute definition and the signaling procedure for backup LSP creation/deletion. It leaves the algorithm of selecting the backup LSPs to maximize backup sharing to carriers/vendors for innovation and value-added enhancements.

2.2 Design Overview

Assume an MPLS network represented by a graph G=(V,E), where V is the set of network nodes[1] (e.g., label switch routers) and E is the set of network links. Protected

[1] In this paper, we use node, router, LSR interchangeably.

LSP requests originating at clients arrive at the source nodes and they are established or deleted in a distributed manner, via signaling among the nodes. For each protected LSP request, the source node needs to compute a service path *Ps* and each node (say node *k*) except the destination node along the service path needs to compute a backup path *Pr(k)* in the network. The backup path *Pr(k)* protects failures of *k*'s immediate downstream node and immediate downstream link. Here *k*'s immediate downstream node is the next node of *k* along *Ps* and *k*'s immediate downstream link is the link between *k* and its immediate downstream node. In general, multiple service LSPs may share a common network resource, hence a single network resource failure, such as link failure or node failure, can cause multiple service LSPs to fail simultaneously. The design objectives for *Ps* and *Pr(k)* are as follows: (1) *Ps* and *Pr(k)* should be *k*'s immediate downstream node disjoint. (2) The required bandwidth associated with different *Pr(k)* of *Ps* should be shared if they share a common link. (3) Backup LSPs from different service LSPs should share bandwidth on common links if their protected service path failure points are not subject to simultaneous failure. (4) Enough bandwidth must be reserved on all links in the network such that for any link/node failure, there is enough bandwidth to restore all affected service LSPs. (5) Total bandwidth reserved for restoration over all links should be minimized. In MPLS networks, a link state routing protocol, such as Open Shortest Path First protocol (OSPF) or Intermediate System-Intermediate System protocol (IS-IS), is used to distribute network topology and resource information. The network resource information includes reserved bandwidth, available bandwidth, and administrative weights on each unidirectional link.

3 Enhancement I

3.1 Bandwidth Management Mechanism

Our bandwidth management mechanism provides a solution to book-keep the bandwidth reserved for restoration on each unidirectional link in a distributed way such that there is only necessary but sufficient bandwidth reserved on each link. For each link, there is bandwidth on both directions. Since each link connects two nodes, we call the direction from smaller node id to larger node id as "up" direction and the reverse direction as "down" direction. Then the node with smaller/larger id is responsible for the bandwidth information of "up"/"down" direction. For each link direction, the responsible node maintains a local array *failother[f]*, where *f* is any possible failure points, denoted as set *F*. In this paper, *f* ranges over any single link and single node, which means $F=\{E\}\cup\{V\}$. *failother[f]* is the amount of bandwidth required on this link direction to restore all failed LSPs if failure *f* occurs. Then, reserving a bandwidth equal to $R = max\ \{Failother[f]: f\in F\}$ ensures that there are sufficient and necessary bandwidth reserved to protect against any single failure. *R* is distributed to all other network nodes using the routing protocol. During the lifecycle of an LSP, the operations include creation and deletion. We describe proposed signaling procedures for updating the local data structures during LSP creation and deletion below.

3.2 LSP Creation

When one source node receives a protected LSP request, it computes a service path *Ps* using constrained shortest path first algorithm (CSPF) with administrative weights and available bandwidth of each unidirectional link. In this paper, we assume the use of the RSVP Traffic Engineering (RSVP-TE) extensions [17] which allow signaling messages to be explicitly routed from the source to the destination. Similar extensions are required for constraint-based routing label distribution protocol (CR-LDP) [18].

LSP creation involves an initial signaling message PATH from the source node to the destination node with admission control along the computed service path, then a capacity reservation message RESV is returned from the destination node to the source node to finish the service path establishment. Upon receiving the RESV message, each node, say *k*, along the service path selects a local repair backup path to the destination node by excluding *k*'s immediate downstream node [2]. Then the node sends a signaling message PATH along the selected backup path to establish the backup LSP and reserve the shared restoration bandwidth. This PATH message will include *k*'s immediate downstream link and k's immediate downstream node information, which is used to maintain shared reservations at each unidirectional link along the backup path. The responsible node of unidirectional link *e* along the backup path updates the array *failother(e)* as follows: *failother(e)[f]* $\leftarrow$ *failother(e)[f]* + *b* where *f* are the immediate downstream link/node along the service path and *b* is the requested bandwidth of the protected LSP.

The implementation of this approach requires a consistent assignment of network link ids and node ids by all network nodes. Since link state routing gives each node a consistent view of the network topology, each link can be indexed by hashing the node ids of the two end points of the link in a standardized way. Another possible solution is for the network provider to provision a globally unique link id for each link.

3.3 LSP Deletion

When a source node receives a LSP deletion request, the network must delete the service LSP and its backup LSPs, and release the reserved bandwidth for shared backup capacity. Using RSVP-TE, the source node sends one PATHTEAR message to the destination node along service path. Each node, say *k*, along the service path also needs to send one PATHTEAR message along each backup path. Those PATHTEAR messages along the backup paths should include their immediate downstream link/node information such that the responsible node of each unidirectional link *e* along the backup path update the array *failother(e)* as follows: *failother(e)[f]* $\leftarrow$ *failother(e)[f]* - *b*, where *f* is the immediate downstream link/node along the service path and *b* is the required LSP bandwidth. Then the reserved bandwidth $R = max \{failother(e)[f]: f \in F\}$ will be updated.

In summary, the updating of *failother* arrays does not require any additional routing and signaling messages. All it needs is to include the immediate downstream node/link information in the PATH and PATHTEAR messages for backup LSPs.

Therefore, the proposed mechanism is very easy to implement in real MPLS networks with fast reroute scheme.

4 Enhancement II

4.1 A Centralized Algorithm with Complete Information

Paper [8] classified the routing information model as minimal information, partial information and complete information in end-to-end restoration. It found that complete information model is able to achieve optimized backup path selection but the complete information dissemination may not be scalable. Thus the complete information is only available for centralized server. In this enhancement, we first describe a centralized algorithm applying complete information to MPLS fast reroute, then we develop a method to realize this algorithm in distributed fashion without complete information flooding in the entire network.

The basic idea of our centralized algorithm is to select the backup LSPs over the links with more sharable bandwidths to reduce the additional required restoration bandwidth. To obtain the sharable bandwidth information, we define a matrix *failneed* from complete information in [8], where *failneed[f,e]* represents the amount of traffic that will be rerouted on unidirectional link *e* when failure *f* occurs. Then the sharable restoration bandwidth for a failure group *Fg* on unidirectional link *e* would be: $r(e) = R[e] - max\{failneed[f,e], f \in Fg\}$, where *R[e]* is the reserved bandwidth on link *e* and provided via routing protocol. In MPLS fast reroute, after the service path *Ps* is selected, we need to select N-1 backup paths starting from each node along the service path except the destination node, where N is the number of nodes along Ps. Let *Pr(k)* be the backup path starting from node *k*. Then *Pr(k)* only requires disjoint from *k*'s immediate downstream link/node along the service path *Ps*, which are denoted as *DL(k)* and *DR(k)* respectively. Then the sharable restoration bandwidth on each unidirectional link *e* for failure group $Fg = \{DR(k),DL(k)\}$ would be: $r(e) = R[e] - max\{failneed[f,e], f \in \{DR(k),DL(k)\}\}$. Next, we reset the link weights according to a function of *r(e)* (see details in section 4.3) and select *Pr(k)* as the shortest path with new link weights. To implement this algorithm on each node, the real challenge is how to collect the information of $max\{failneed[f,e], f \in \{DR(k),DL(k)\}\}$. Note that each node does not need the complete matrix *failneed*, but the rows corresponding to its immediate downstream link/node failures. Here we propose using signaling protocol to collect the required information on each node. Our proposal keeps signaling overhead very small.

4.2 Signaling Extensions

As *failother* array in section 3.1, we propose that for each failure, a master node is responsible for maintaining the failure related information. For link failure, the master node could be the node terminating the link having the smaller node id. For node failure, the master node would be the node itself. Then the master node of each failure *f* along the service path maintains a *failself(f)* array. *failself(f)[e]* stores the bandwidth required on unidirectional link e to restore all affected LSPs if this failure f occurs. Note that *failself(f)* corresponds to the failure *f* row in the matrix *failneed* defined above.

We propose three new signaling messages between neighbor LSRs: TELL, ASK and REPLY for updating and obtaining the *failself* arrays.

- TELL: after a node selects or deletes its backup path, it will send a TELL message to its immediate downstream node along service path with backup path information.
- ASK: before a node selects its backup path, it sends a ASK message to its immediate downstream node for information.
- REPLY: when a node receives the ASK message, it replies the correct information back to the ask node.

The new message operates as follows: The ASK and REPLY messages are used for obtaining the *failself* array information. Before a node selects its backup path, it will send a ASK message to its immediate downstream node for *failself* array information for the immediate downstream node and/or the immediate downstream link. The REPLY message will reply the correct *failself* arrays of the immediate downstream node and the immediate downstream link information if it is the master node of the link.

The TELL message is used for updating *failself* arrays when a backup LSP is setup or teardown. After a node creates or deletes a backup LSP protecting the immediate downstream node/link failures, it sends a TELL message including backup LSP information to the immediate downstream node. Upon receiving the TELL message the immediate downstream node will update the *failselfs* array for itself and the immediate downstream link if it is the master node of the downstream link: *failself[e]* ← *failself[e]* ± *b*, where *e* are the unidirectional links on the restoration LSP, *b* is the bandwidth of the LSP and "+" is for creation, "-" for deletion.

Since the *failself* arrays for the immediate downstream node/link failures are either installed in the immediate downstream node or itself, a node can easily obtain the *failself* array information needed to select the backup LSP path same as the centralized server. Therefore, we have successfully realized the centralized optimal backup path selection algorithm in a distributed fashion.

4.3 Algorithm in Detail

After a node *k* along the service path collects the *failself* array(s) from its immediate downstream node for immediate downstream link/node failures, it calculates a new array *T[e] = max(failself(DR(k))[e],failself(DL(k))[e])*, where *e* is for any unidirectional link. Then *T[e]* is the maximum bandwidth needed on unidirectional link *e* if any one of the protected failures *DR(k)/DL(k)* occurs, and the sharable backup capacity on *e* is *r[e] = R[e] - T[e]*, where R[e] is the reserved bandwidth on link e. After that, the node assigns a new weight to each unidirectional link *e* in the network:

$$w[e] = \begin{cases} (b - r[e]) \cdot W[e] & \text{if } b - r[e] > 0 \text{ and } e \notin \{DR\} \\ \varepsilon & \text{if } (b - r[e] \le 0 \text{ or } e \in \{DP\}) \text{ and } e \notin \{DR\} \\ \infty & \text{if } e \in \{DR\} \text{ or } \mathrm{A}[e] < b - r[e] \end{cases}$$

where *{DR}* means the set of links that adjacent to the immediate downstream node, which means the backup path should exclude this immediate downstream node, *{DP}* means the set of links downstream from the immediate downstream node along the service path, *W[e]* is the administrative weight on unidirectional link *e, A[e]* is the available bandwidth on unidirectional link *e,* and ε is a very small positive number. Then shortest path algorithm is used to select the backup path using these weights. In this formula, the assigned new weight of ε is in favor of the selection of the links, which are the downstream path links for potential path merging and links with enough sharable restoration capacity. In addition, if a link does not have enough available capacity *A[e]*, it should be excluded from the backup path selection as well.

4.4 Scalability

The proposed scheme scales well due to two facts: (1) it does not require each LSR to maintain the global information of per LSP's service path and restoration paths. Only two arrays, i.e., *failself* and *failother*, are maintained at nodes independently. (2) it does not require each node to broadcast any additional messages in the entire network. The signaling overhead per backup LSP is confined between two adjacent nodes. Although we introduced three messages, in real implementation, only the TELL message needs to be added. The ASK and REPLY functionality can be embedded in the PATH and RESV messages as below. During the service LSP setup process, we use an optional field in PATH message to request the *failself* information. When RESV message comes back from destination to source, the *failself* array is carried in an optional field from downstream node to the immediate upstream node. Hence the only signaling overhead in our proposal would be a single TELL message between two adjacent nodes per backup path.

5 Study Methodology

We evaluate the performance of our proposed approaches via simulation on two networks: the first is a US MPLS backbone network consisting of 18-PoPs (Points-of-Presence) and 30 links interconnected in a dual backbone router (BR) architecture[2] [19]; the second is the Toronto metropolitan network consisting 25 nodes and 45 links from paper [12]. In the first network, we aggregate all access routers (ARs) connecting to the same PoP into a virtual AR. Then each PoP consists of three nodes: two BRs and one AR. The AR-to-AR demands are generated using cumulative outgoing and incoming traffic from a demand forecast, i.e., all traffic starting and terminating at the AR respectively. Then we randomly select a demand according to source AR

[2] For router protection, IP/MPLS service providers may use two backbone routers at each PoP and each access router is dual-homed to the two backbone routers.

cumulative traffic outgoing probability and destination AR cumulative traffic incoming probability. In the second network, we do not have traffic forecast demands and thus assume uniform distribution traffic: randomly select two nodes for demand source and destination. The bandwidth of each demand is uniformly generated from an interval (10,100) Mbps for both networks. For simplicity, the MPLS link bandwidths are assumed to be deployed in units of OC48 channels (approx. X=2.5 Gbps). In our simulation, we assume traffic demands arrive at the network one by one, and never leave. In our simulation results, we generate 20 random runs and calculate the average value for each data point. We evaluate the performance of our proposals using restoration overbuild. We define α=ΣSC(i) and β=ΣTC(i), where i varies over all links, SC(i) and TC(i) are the required number of OC48 channels for service only and for both service and restoration on link i respectively. Since service and restoration share the same bandwidth pool, it is possible SC(i)=TC(i) on some links. Then the restoration overbuild is calculated as (ΣTC(i)-ΣSC(i))/ΣSC(i).

6 Simulation Results

We assume that the link capacities are set to infinity and LSP demands arrive one at a time. The service path is always routed along the shortest path. The backup paths will be selected using immediate downstream node disjoint shortest path to the destination node in baseline and our enhancement I while the backup paths in enhancement II are selected using our new link weight setting. Only merging technique is used in baseline scheme while distributed bandwidth management is used in both enhancements. After all LSPs are routed on the network, we calculate the restoration overbuild for each of the schemes.

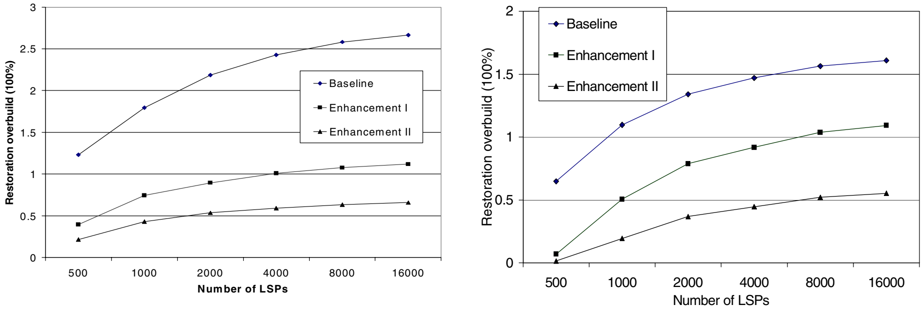


Fig. 2. Network Restoration Overbuild

Figure 2 illustrates the average restoration overbuild for each of the three schemes. The left side sub-figure shows the restoration overbuilds for the first network and the right side sub-figure for the second network. Each algorithm is compared with the same traffic demands ranging from 500 to 16000 protected LSPs per network. Obviously, both enhancements require significantly fewer restoration overbuilds than the baseline scheme and our enhancement II requires the least for each demand set. Those conclusions are not surprising. Since baseline only uses merging technique and only shares bandwidth between the protected LSP service path and its backup paths, then each protected LSP reserves restoration bandwidth individually. Thus the baseline scheme reserves restoration bandwidth comparable to dedicated 1+1 restoration, which is very bandwidth consuming. Our enhancement I uses the default backup path selection but provides bandwidth sharing among any LSPs, it reserves the restoration bandwidth comparable to simple disjoint shortest path restoration with bandwidth sharing. Our enhancement II uses a little bit extra information to select the optimal backup path to maximize backup capacity sharing, it consumes the least restoration overbuild and it reserves the restoration bandwidth comparable to optimized shared mesh restoration with complete information [7,8,10].

We notice that the higher the traffic demands, the larger the restoration overbuilds in our simulation results. This is due to the restoration overbuild calculation formula. If the bandwidth in service channels is not fully consumed by service paths, it will be used for restoration paths. Fewer demands could leave more bandwidth on service channels for restoration paths. When the number of LSPs increases, the overbuild will stabilize. Note that enhancement I performs more than 100% and about 50% better than the baseline scheme in restoration overbuild for the first network and the second network respectively. Our enhancement II requires about close to half overbuild than enhancement I in both networks. Thus, there are strong advantages to use our proposed enhancements for MPLS fast reroute. All our enhancements use distributed operation and no centralized server is required. Also we noticed that the baseline scheme performs better in the second network than in the first one. One potential reason could be that the second network is denser than the first one. Using our enhancements, the restoration overbuilds are nearly the same for both networks. We also simulated blocking probability of the three fast reroute schemes on both networks and found out that both enhancements are able to reduce blocking probability significantly from baseline and enhancements II always performs the best. Due to page limits, we left out the blocking probability simulation results.

7 Conclusion

We analyzed the problem of distributed bandwidth sharing and backup path selection for protected LSPs in MPLS fast reroute using one-to-one backup method. In particular, we proposed a distributed bandwidth sharing mechanism with implementation procedures. We also proposed a new distributed algorithm, which relies on limited routing information to achieve complete routing information results using simple signaling extensions between neighbor LSRs to distribute and collect extra information. We demonstrate that with small signaling overhead, we can use network capacity much more efficiently for MPLS fast reroute. We compared our new approaches against the IETF Internet draft [2] proposed solution. We conclude that the savings from our two enhancements are significant, while the implementation overhead is only marginal. Future work will focus on facility backup method of MPLS fast reroute.

References

[1] E. Rosen et al. (ed.), "Multi-protocol Label Switching Architecture," IETF RFC 3031, 2001

[2] P. Pan, G. Swallow, and A. Atlas (Editors), "Fast Reroute Extensions to RSVP-TE for LSP Tunnels," IETF Internet Draft, draft-ietf-mpls-rsvp-lsp-fastreroute-05.txt, 2004, working in progress..

[3] S. Kini et al., "Shared Backup Label Switched Path Restoration", IETF Internet draft, work in progress, May 2001.

[4] V. Sharma and F. Hellstrand (ed.), "Framework for Multi-Protocol Label Switching MPLS-based Recovery," IETF RFC 3469, 2003.

[5] C. Gruber, "Bandwidth Requirements of MPLS Protection Mechanisms," 7th INFORMS TELECOM, Boca Raton, Florida, 2004.

[6] G. Li and D. Wang, "Efficient Restoration Capacity Design in MPLS networks," APCC/MDMC, Beijing, China, 2004.

[7] G. Li, D. Wang, C. Kalmanek and R. Doverpike, "Efficient Distributed Path Selection for Shared Restoration Connections", IEEE Infocom, New York, Vol. 1, pp.140-149, 2002.

[8] M. Kodaliam and T. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration", IEEE INFOCOM 2000.

[9] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating Optimal Space Capacity Allocation by Successive Survivable Routing," IEEE INFOCOM 2001.

[10] C. Qiao and D. Xum "Distributed Partial Information Management (DPIM) for Survivable Networks," INFOCOM 2002.

[11] R. Doverspike and J. Yates, "Challenges for MPLS in Optical Network Restoration," IEEE Communication Magazine, Feb. 2001.

[12] Y. Xiong and L. Mason, "Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks," IEEE/ACM Transactions on Networks, vol 7(1), February 1999.

[13] A. Basu and J. Riecke, "Stability Issues in OSPF Routing," SIGCOM 2001.

[14] C. Alaettinoglu, V. jacobson, and H. Yu, "Towards Millisecond IGP Convergence," Internet draft, draft-alaettinoglu-isis-convergence-00.txt, IETF, Nov. 2000.

[15] D. Katz et al., "Traffic Engineering (TE) Extensions to OSPF Version 2," RFC 3630, Sept. 2003.

[16] H. Smit et al., "Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)," RFC 3784, June 2004.

[17] D. Awduche et al, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC3209, Dec. 2001.

[18] J. Ash et al, "LSP Modification Using CR-LDP," RFC 3214, Jan. 2002.

[19] G. Li, et al, "Detail Study of IP/ Re-configuable Optical Network Architectures," Broadnets 2004.

[20] P. Ho, J. Tapolcai, and H. Moufth, "On achieving optimal survivable routing for shared protection in survivable next-generation internet", IEEE Transaction on Reliability, Vol 53(2), June 2004, pp216-225.

[21] Pin-Han. Ho, "Segment Shared Protection in Mesh Communications Networks with Bandwidth Guaranteed Tunnels," to appear IEEE/ACM ToN.

[22] P. Ho and H. Mouftah, “Reconfiguration of spare capacity for MPLS-based recovery for the Internet Backbone networks,” IEEE/ACM ToN, Vol 12(1), Feb. 2004, pp 73-85.

[23] S. Han and G. Shin, "Fast restoration of real-time communication service from component failure in multi-hop networks," ACM SigComm 1997.

[24] D. Wang et al, “Efficient segment-by-segment restoration,” OFC 2004.

Pricing for Heterogeneous Services at a Discriminatory Processor Sharing Queue

Yezekael Hayel[1] and Bruno Tuffin[1]

IRISA-INRIA Rennes, Campus universitaire de Beaulieu,
35042 Rennes Cedex - France
Tel: (+33)299847134, Fax: (+33)299842529
{yhayel, btuffin}@irisa.fr

Abstract. In order to deal with applications with different quality of service requirements, service differentiation has to be implemented, especially in case of congestion. Different scheduling policies can be applied at a queue, such as strict priorities, generalized processor sharing, or discriminatory processor sharing. While prices optimizing the network revenue have been determined in the first two above cases, and the optimal revenue compared, nothing had been done yet on discriminatory processor sharing (DPS). Though, at the session level, processor sharing is known to properly model TCP flows behavior. DPS then models multiple TCP flows at a router providing differentiated services. We study here what pricing induces on a DPS router when two types of application compete for service, what is the resulting equilibrium, and explain how optimal prices can be found.

Keywords: Economics, Queueing theory.

1 Introduction

Telecommunication networks such as the Internet are facing congestion due to increasing demand in terms of number of subscribers as well as in terms of more and more demanding applications. For this reason, quality of service (QoS) requirements may not be satisfied by the current best-effort service. To cope with this problem, architectures have been designed based on the idea of treating flows differently. IntServ [1] architecture applies resource reservation but is not scalable. DiffServ architecture [2] has then been designed to tackle this scalability problem. It consists in defining different classes of service (by marking packets with a DiffServ codepoint (DSCP)) and applying a given scheduling policy for serving the packets of the different classes at routers (according to the DSCP).

Surprisingly, no pricing procedure has been associated to the defined architectures, at least in the process of their design. However, considering free services or flat-rate pricing, selfish users would send all their packets to the class providing the "best" quality of service, so that congestion would persist. Pricing for network usage and/or congestion has recently received a lot of attention (see [3]

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 816–827, 2005.

and the references therein), but some of those works are not devoted to multi-class pricing like, among the most noteworthy schemes, auctions for bandwidth (see for instance [4]), or charging for elastic traffic based on transfer rate [5, 6]. In the context of multi-class pricing, excepted pricing for guaranteed services [7], most papers deal with pricing for strict priority [8, 9] due to its applicability. Nevertheless, priority queueing is not the only scheduling policy likely to be implemented in DiffServ architecture, and the good choice for a provider may be related to the financial benefits that it provides.

Based on this idea, we have compared in a previous paper [10] the respective impact of generalized processor sharing (GPS) and priority queueing (PQ) on the provider's revenue, when the QoS parameter studied is the delay. The model was taken from [9] where priority queueing was considered. Since no closed-form expression is available for the delay at a GPS queue [11], an approximation by separate queues has been used, for which the delay is known. This approximation is valid in the context of congested queues, which is actually the situation where service differentiation is relevant. We have especially proved that, at least for the studied class of utility functions and when modelling the router by an M/M/1 queue, PQ is the policy that always yields the highest revenue, and should then be preferred.

We wish to study in the present paper the impact of another scheduling policy, namely discriminatory processor sharing (DPS), on users' behavior and on the provider's revenue. DPS provides a more flexible way of giving priority than the PQ discipline. It has been introduced by Kleinrock [12] for a single server. It consists in serving classes in proportions controlled by weights, while packets within a given class are served according to a standard processor sharing strategy (which constitutes the main difference with respect to GPS, where packets within a class are served according to a FIFO scheme) [13]. A theoretical charm of DPS is that, unlike GPS, a closed-form solution of steady-state delay exists [14]. DPS has been justified in practice, at the flow level, as a fluid approximation of some weighted round-robin scheme [15]. It has also been shown in [16, 17] that a queue with DPS discipline is well adapted, as an approximation, to the way TCP connections can share bandwidth. It uses the fact that, at the session level, TCP can be analyzed using a PS approach [18]. We thus use DPS at the flow level here in order to fit the TCP modelling.

The contribution of the paper is as follows. We consider two types of applications applying for service at two classes of service. In our analysis, we focus on dedicated classes, that is the case where a given application is directed to a given class of service. We consider that the two types of application both use TCP traffic, so that a DPS queue can be used to approach the system behavior. The modelling of TCP constitutes one of the main enhancements with respect to [9, 10]. We are going to investigate the impact of DPS on the pricing model. Since, with respect to PQ and the approximation of GPS, delay over classes are *mutually* dependent, a deeper analysis of the model at the infinitesimal level is required. We show that, for fixed prices, there always exists a unique equilibrium for the average number of sessions of each type competing for service. This equilibrium is then a so-called Nash equilibrium often used in micro-economics studies [3], where no user/application class has any incentive to unilaterally deviate from its current policy. We show that, depending on the price values, we may have undesirable situations with only one class of traffic asking for service; a necessary and sufficient condition for avoiding that is provided. We then describe how optimal prices can be found and illustrate that PQ is (still) the border case maximizing the provider's revenue among all discriminatory processor sharing parameter choices.

The rest of the paper is organized as follows. In Section 2, we present the model, mainly taken from [9], and adapt it to TCP with DPS scheduling policy. Section 3 then discusses in details the equilibrium that can obtained depending on the prices selection. Section 4 discusses how optimal prices can be found and Section 5 illustrates numerically the previous results. Finally, Section 6 concludes the paper and gives some perspectives of research.

2 Model

Our general model closely follows [9]. We assume that service is differentiated by two classes of traffic. We also assume that there are two types of users/applications, delay-tolerant users that we will abusively (but for sake of simplicity) call data users, and delay-sensitive users, that we will call voice-users. We keep the notations voice and data to be consistent with [9, 10], but, especially as we consider TCP traffic, voice can/should be replaced by any application with low delay preferences. In this paper, we will force data users to one class and voice users to the other class. This is called the case of *dedicated* classes in [9]. The case of *open* classes where the users of each application can choose between classes is left for future work.

We consider a system with infinite population of potential customers applying for service as soon as the *cost* of sending packets is less than the benefits that they get from it. The benefits of sending packets are expressed by *utility* functions, $u_d(\cdot)$ for data and $u_v(\cdot)$ for voice, which depend on the mean delay in each class respectively denoted by $\mathbb{E}D_d$ and $\mathbb{E}D_v$. We consider the utility as a function of the average delay rather than the average utility of instantaneous delay since average delay seems to us the measure of interest, especially when dealing with data for instance (even for voice, average delay seems more relevant provided that an upper-threshold is not reached). Like in [9, 10], we consider utility functions $u_d(y) = y^{-\alpha_d}$ and $u_v(y) = y^{-\alpha_v}$ with $0 < \alpha_d < \alpha_v$, so that voice users value low delay more than data users, but conversely value high delay less. This choice of utility function is somewhat arbitrary, but is based on the idea that utility curves do intersect [9]: low delay is more valuable to voice than data, and conversely large delay is more valuable to data. Note also that this form of utility function can be related to the Cobb-Douglas function widely used in micro-economic analysis [19].

Let p_d and p_v be the per-packet price for access at respectively data and voice classes. The *residual utility* of a data user (resp. voice user) is given by

$$u_d(\mathbb{E}D_d) - p_d \quad \text{(resp.} \quad u_v(\mathbb{E}D_v) - p_v). \tag{1}$$

Sessions open as Poisson arrival processes. Let N_d and N_v be the mean number of data and voice users sending packets in steady-state. Let also λ_d and λ_v be the average size of data and voice flows (drawn independently between flows). In [9, 10], N_d and N_v were the *fixed* numbers of sources while λ_d and λ_v were the rates at which packets were sent. The model is then here slightly different in order to fit the usual assumptions for modelling TCP traffic [16, 17, 18] by using processor sharing. Even if using λ for the average size is unusual, we keep the notation here to be consistent with the notations in [9, 10]. As we consider Poisson flow arrivals, the number of flows in progress behaves like the number of customers in an M/M/1 processor sharing queue [12]. Moreover, as we assume two different class of traffic, the model behaves like an M/M/1 discriminatory processor sharing queue. Let μ be the service rate of the server/router.

We consider a discriminatory processor sharing (DPS) scheduling policy for differentiating services. DPS basically works as follows. There exists a nonnegative parameter γ representing *relative priority* of data customers and $1-\gamma$ for voice customers. Still, when packets of one class are not present in the queue, the server is fully allocated to the other class, but flows within a class are served according to a processor sharing (PS) scheme. A closed-form formula for the average delays in such M/M/1 queues are given in [20–page 86] by

$$\mathbb{E}D_v = \frac{\left(1 + \frac{\lambda_d N_d(2\gamma-1)}{\mu-(1-\gamma)\lambda_v N_v - \gamma\lambda_d N_d}\right)}{\mu - \lambda_v N_v - \lambda_d N_d} \quad \text{and} \quad \mathbb{E}D_d = \frac{\left(1 - \frac{\lambda_v N_v(2\gamma-1)}{\mu-(1-\gamma)\lambda_v N_v - \gamma\lambda_d N_d}\right)}{\mu - \lambda_v N_v - \lambda_d N_d}. \tag{2}$$

Remark that PQ (but with PS discipline within a class) is a special case of DPS, since $\gamma = 0$ gives strict priority to voice users, leading to expression of delay:

$$\mathbb{E}D_v = \frac{1}{\mu - \lambda_v N_v} \quad \text{and} \quad \mathbb{E}D_d = \frac{\mu}{(\mu - \lambda_v N_v)(\mu - \lambda_v N_v - \lambda_d N_d)},$$

while $\gamma = 1$ would give strict priority to data (note that the mean delay is the same for M/M/1/FIFO and M/M/1/PS queues).

3 Equilibrium Analysis

Whereas determining the (mean) number of users of each type in equilibrium is quite easy for PQ and an approximation of GPS where the queues are logically separated, the analysis is much more intricate for DPS due to the influence of the number of users of one type on the delay of the other type of users, and vice versa.

The steady-state average numbers of sessions of each type has to ensure that mean residual utilities (1), are positive or null, otherwise the number of sessions naturally decreases. Following the same line, this average number increases until the residual utility approaches 0. It means that each type of application naturally adapts its steady-state number of sources in the sense the maximum (mean) number of sources of a given type cannot make negative its residual utility. We hence have a game between the different types of applications, for the maximum mean number of sessions, potentially leading to a Nash equilibrium. Let us now investigate the existence and uniqueness of this equilibrium.

Let

$$U_d(N_d, N_v) = u_d(I\!\!E D_d) - p_d \quad \text{and} \quad U_v(N_d, N_v) = u_v(I\!\!E D_v) - p_v$$

be the mean residual utilities, where we emphasize the role of the number of connections of each type. We obtain that N_v and N_d must verify

- If $N_d, N_v > 0$, then $U_d(N_d, N_v) = U_v(N_d, N_v) = 0$,
- If $N_d > 0,\ N_v = 0$, then $U_d(N_d, N_v) = 0,\ U_v(N_d, N_v) \leq 0$,
- If $N_d = 0,\ N_v > 0$, then $U_d(N_d, N_v) \leq 0,\ U_v(N_d, N_v) = 0$.

The last two relations mean that the type of application such that $N = 0$ has no incentive to open sessions since its mean residual utility is negative.

Figure 1 illustrates these equations, where the maximum number of customers of each type in the network is necessarily on the "minimum" of curves $U_d(N_d, N_v) = 0$ and $U_v(N_d, N_v) = 0$. Indeed, the mean number of sources of each type increases, decreasing then the utilities, until a residual utility reaches zero. Figure 1 depicts the three situations that will be used later on: either the curves cross each other in the domain $\{(N_d, N_v) : N_d, N_v \geq 0\}$, or one curve is always under the other in this domain.

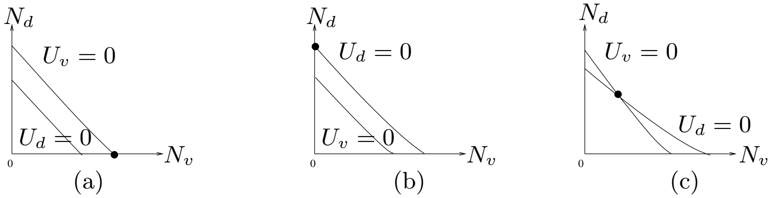


Fig. 1. Curves for the (maximum) number of customers of each type, and the associated Nash equilibrium. The three possible cases are displayed. The resulting Nash equilibrium is displayed by the thick point

Lemma 1. *Due to the form of utility functions, the curves* $U_d(N_d, N_v) = 0$ *and* $U_v(N_d, N_v) = 0$ *either are one above the other or cross each other only once on* $N_d, N_v \geq 0$.

This lemma is proved in Appendix 1.

Wherever this (minimal) curve such that one type of application has reached his null (residual) utility, the system may continue to evolve since if the utility of the other class is positive, it will continue to increase its number of source, increasing then the delay of the first class (see Equations (2)), so that its utility will become negative, meaning that the number of sources will have to be reduced.

To simplify the notations, let $q_d = (p_d)^{1/\alpha_d}$ and $q_v = (p_v)^{1/\alpha_v}$. The following proposition establishes the existence of a unique Nash equilibrium, and gives an explicit solution in terms of q_d and q_v.

Proposition 1. *Consider fixed prices p_d and p_v. There exists a unique (Nash) equilibrium (N_d^*, N_v^*) that stabilizes the mean number of emitting sources.*

- *If the curve $U_d(N_d, N_v) = 0$ is always under $U_v(N_d, N_v) = 0$ on $N_v, N_d \geq 0$ (case (a) of Figure 1), the Nash equilibrium is given by*

$$(N_d^*, N_v^*) = \left(0, \frac{\mu - q_v}{\lambda_v}\right). \tag{3}$$

- *If the curve $U_v(N_d, N_v) = 0$ is always under $U_d(N_d, N_v) = 0$ on $N_v, N_d \geq 0$ (case (b) of Figure 1), the Nash equilibrium is given by*

$$(N_d^*, N_v^*) = \left(\frac{\mu - q_d}{\lambda_d}, 0\right). \tag{4}$$

- *If the curves have a crossing point (case (c) of Figure 1), it is unique, and its value is a Nash equilibrium $(N_d^*, N_v^*) > 0$, with*

$$\begin{aligned}\lambda_d N_d^* &= \frac{\mu q_d}{\gamma q_d - (1-\gamma) q_v} - \frac{q_v q_d}{\gamma q_v - (1-\gamma) q_d}\\ \lambda_v N_v^* &= -\frac{\mu q_v}{\gamma q_d - (1-\gamma) q_v} + \frac{q_v q_d}{\gamma q_v - (1-\gamma) q_d}.\end{aligned}$$

We will call this last case the non-trivial equilibrium.

The proof of this proposition is given in Appendix 2.

An important remark is that the two border equilibriums (3) and (4) correspond to the cases where only one class of traffic eventually requests service. It then means that the model is a queue with one class of service, and no real service differentiation is applied (except that the other class is rejected). As a consequence, in order to obtain a gain by differentiating service among users, we have to look at prices under which the above border equilibriums are not reached.

The following proposition gives a necessary and sufficient condition over prices for being in a non-trivial equilibrium.

Proposition 2. *The Nash equilibrium is non-trivial if and only if q_v and q_d satisfy*

$$0 < q_v < \mu, \tag{5}$$

$$\sqrt{\mu q_v + (\frac{\mu - q_v}{2}\frac{1-\gamma}{\gamma})^2} - \frac{\mu - q_v}{2}\frac{1-\gamma}{\gamma} < q_d, \tag{6}$$

$$\frac{\mu\gamma q_v + (1-\gamma) q_v^2}{\gamma q_v + \mu(1-\gamma)} > q_d. \tag{7}$$

The proof of this proposition is given in Appendix 3.

Remark: It can be verified with straightforward calculus that the traffic load ρ is less than one (which is the stability condition for the queue).

4 On Optimal Prices and Revenues

The idea, from a network provider's perspective, is to find the p_d^*, p_v^* and γ^* giving the maximum network revenue

$$R^* = \max_{p_d, p_v, \gamma} \lambda_d N_d^*(p_d, p_v, \gamma) p_d + \lambda_v N_v^*(p_d, p_v, \gamma) p_v$$

subject to $p_v, p_d \geq 0$ and $\gamma \in [0, 1]$.

It is possible to analytically determine the optimal γ for fixed values of p_d and p_v, but optimizing then in terms of p_d and p_v looks analytically intractable, so that the determination of γ is not helpful (particularly since it requires a decomposition in several cases).

In the next section we present numerical results illustrating the domains for border or non-trivial equilibrium, as well as the impact of prices and bandwidth partition on the network provider's revenue. The prices optimizing the revenue are also numerically determined.

5 Numerical Illustrations

In this section we will present some numerical results obtained with the above model[1]. Unless stated otherwise, the following parameters will be used throughout this section: $\mu = 1$, $\alpha_v = 1.8$ and $\alpha_d = 1.5$, $\lambda_v = 0.1$ and $\lambda_d = 0.3$.

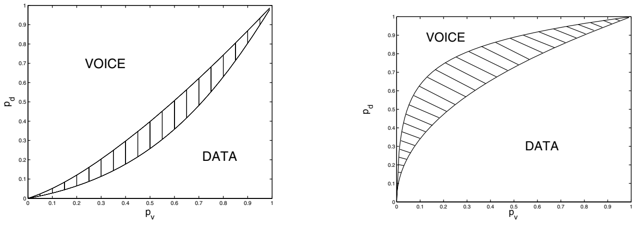
5.1 Price Domains and Associated Equilibrium

Figure 2 illustrates the equilibrium domains in terms of prices for two values of γ, say $\gamma = 0.9$ and $\gamma = 0.2$. We have the three possible areas: VOICE corresponding to the (border) equilibrium where there is only voice traffic, DATA corresponding to the case where there is only data traffic, and the hatched area corresponding to a non-trivial equilibrium. Note that when γ is less than 0.5, the zone for non-trivial equilibrium is such that $p_v^{1/\alpha_v} > p_d^{1/\alpha_d}$, and that the converse is also true.

5.2 Optimal γ for Fixed Prices

For given prices p_v and p_d, it might be interesting to look at the bandwidth-sharing parameter γ optimizing the revenue. This value is shown on Figure 3(a). It can be observed that when p_v is large and p_d is small, the optimal revenue is obtained when $\gamma^* = 0$, i.e. a strict priority is assigned to voice traffic, and conversely when p_v is low and p_d is large, $\gamma^* = 1$, meaning that a strict priority should be applied to data traffic.

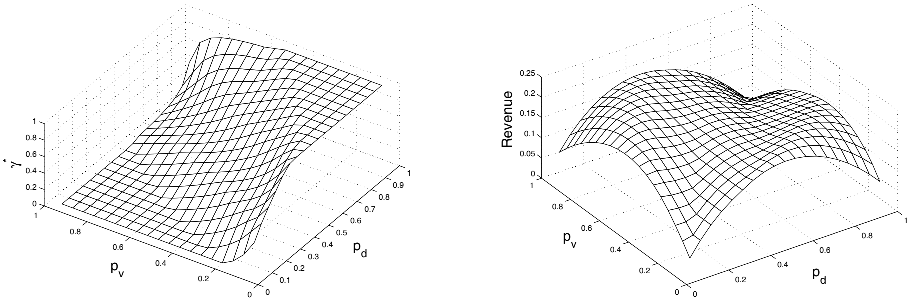
[1] The standard numerical and optimization packages of any mathematical software can be used to solve this problem.



(a) $\gamma = 0.2$

(b) $\gamma = 0.9$

Fig. 2. Equilibrium domains with different values of parameter γ and varying prices



(a) Optimal bandwidth-sharing

(b) Provider's revenue

Fig. 3. Optimal bandwidth-sharing and provider's revenue in terms of prices

5.3 Revenue

Figure 3(b) shows the revenue in terms of prices, using the optimal value of γ used for each couple (p_d, p_v). Since the curve of revenue is smooth, obtaining the optimal prices is numerically easy. For this example, we obtain the optimal prices $p_d^* = 0.24$ and $p_v^* = 0.68$, giving an optimal revenue $R^* = R(p_d^*, p_v^*, 0) = 0.21$.

It is important to note that, at optimal prices, the optimal γ is then 0, giving then strict priority to voice traffic. Actually, whatever the choices of parameters we have, this result has been verified by extensive simulations that cannot be reported here for lack of room. This is somehow in accordance with [10] where we have shown for a simple M/M/1 queue that PQ has to be preferred to GPS to optimize the revenue. In the present case, a formal proof of this conjecture has still to be found.

6 Conclusions and Perspectives

We have studied in this paper the impact of pricing on a multi-class queue with DPS scheduling policy (known to properly model service differentiation between

TCP flows). We have shown, in the case of two types of applications and of dedicated classes that the game between both types of traffic for setting their mean number of emitting sources always results in a unique Nash equilibrium. The location of this equilibrium depends on prices. We have determined three domains of prices for which either only voice requests service, either only data requests service or for which traffic is mixed (the so-called non-trivial equilibrium). We have numerically investigated the price selection in order to optimize the network revenue and illustrated that PQ does actually seem to always provide a higher revenue.

As directions for future work, we wish to prove theoretically that PQ is the limit case of DPS always providing the highest revenue, and should then be preferred, as done in [10] for GPS. Studying the behavior of other scheduling policies is also a topic of interest, in order to define the best strategy for optimizing the network's revenue. We also wish to compare/verify our results with ns-2 simulations.

References

1. Braden, R., Clark, D., Shenker, S.: Integrated Services in the Internet Architecture: an Overview. Technical report, RFC 1663 IETF (1994)
2. Wang, Z.: Internet QoS: Architectures and Mechanisms for Quality of Service. Morgan Kaufmann Publishers (2001)
3. Courcoubetis, C., Weber, R.: Pricing Communication Networks—Economics, Technology and Modelling. Wiley (2003)
4. Maillé, P., Tuffin, B.: Multi-bid auctions for bandwidth allocation in communication networks. In: Proceedings of IEEE INFOCOM. (2004)
5. Kelly, F., Mauloo, A., Tan, D.: Rate control in communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research Society **49** (1998) 237–252
6. Low, S., Lapsley, D.: Optimization Flow Control, I: Basic Algorithm and Convergence. IEEE/ACM Transactions on Networking **7** (1999)
7. Paschalidis, I., Liu, Y.: Pricing in Multiservices Loss Networks: Static Pricing, Asymptotic Optimality, and Demand Substitution Effects. IEEE/ACM TON **10** (2002)
8. Cocchi, R., Estrin, D., Shenker, S., Zhang, L.: Pricing in Computer Networks: Motivation, Formulation and Example. IEEE/ACM TON **1** (1993) 614–627
9. Mandjes, M.: Pricing Strategies under Heterogeneous Service Requirements. In: Proceedings of IEEE INFOCOM. (2003)
10. Hayel, Y., Ros, D., Tuffin, B.: Less-than-Best-Effort Services: Pricing and Scheduling. In: Proceedings of IEEE INFOCOM. (2004)
11. Borst, S., Boxma, O., Núnez Queija, R.: Heavy Tails: The Effect of the Service Discipline. In: Proceedings of Performance TOOLS. (2002)
12. Kleinrock, L.: Time-shared Systems: A Theoretical Treatment. Journal of the Association for Computing Machinery **14** (1967)
13. Rege, K., Sengupta, B.: Queue-Length Distribution for the Discriminatory Processor-Sharing Queue. Operations Research **44** (1996)
14. Fayolle, G., Mitrani, I., Iasnogorodski, R.: Sharing a Processor Among Many Job Classes. Journal of the Association for Computing Machinery **27** (1980)

15. Mitra, D., Weiss, A.: A Closed Network with a Discriminatory Processor-Sharing Server. In: Proceedings of ACM Sigmetrics. (1989)
16. Altman, E., Jimenez, T., Kofman, D.: DPS Queues with Stationary Ergodic Service Times and the Performance of TCP in Overload. In: Proceedings of IEEE INFOCOM. (2004)
17. Bu, T., Towsley, D.: Fixed point approximations for TCP behavior in an AQM network. In: Proceedings of ACM Sigmetrics. (2001)
18. Ben Fredj, S., Bonald, T., Proutiere, A., Regnie, G. Roberts, J.: Statistical Bandwidth Sharing: a Study of Congestion at Flow Level. In: Proceedings of IEEE SIGCOMM'. (2001)
19. Beckert, W.: Stochastic Demand Analysis. PhD thesis, UC Berkeley (2000)
20. Hassin, R., Haviv, M.: To Queue or Not To Queue. Kluwer's (2003)

Appendix

1 Proof of Lemma 1

Look at the non-linear system defined by $U_v(x,y) = U_d(x,y) = 0$ and solve it over $I\!R^2$. It can be rewritten as

$$\begin{cases} \frac{\mu - x\lambda_v - y\lambda_d}{q_v} = 1 + \frac{y\lambda_d(2\gamma-1)}{\mu-(1-\gamma)x\lambda_v - y\gamma\lambda_d} \\ \frac{\mu - x\lambda_v - y\lambda_d}{q_d} = 1 - \frac{x\lambda_v(2\gamma-1)}{\mu-(1-\gamma)x\lambda_v - y\gamma\lambda_d} \end{cases}$$

The unique solution (x^*, y^*) is:

$$x^* = \left(\frac{\mu q_d}{\gamma q_d - (1-\gamma)q_v} - \frac{q_v q_d}{\gamma q_v - (1-\gamma)q_d}\right)\frac{1}{\lambda_d},$$

$$y^* = \left(-\frac{\mu q_v}{\gamma q_d - (1-\gamma)q_v} + \frac{q_v q_d}{\gamma q_v - (1-\gamma)q_d}\right)\frac{1}{\lambda_v}.$$

Equations $U_d(x,y) = 0$ and $U_v(x,y) = 0$ both actually give y as a function of x since to each x corresponds only one y. These functions are continuous so that either one curve is always above the other on the possible domain of (N_d, N_v), either the curves cross each other only once from the unique solution of the system. □

2 Proof of Proposition 1

Consider the dynamics of the queue for the three possible curves configurations.

- If the curve $U_d(N_d, N_v) = 0$ is always under $U_v(N_d, N_v) = 0$ on $N_v, N_d \geq 0$ (with the condition that the queue is ergodic), (case (a) of Figure 1), the application types will increase their mean number of emitting sessions until a residual utility function approaches 0, that is U_d in the present case. The system continues then to evolve and only voice traffic increases its number of sources because its utility function is still strictly positive. This implies that the number of data sessions will naturally decrease as its utility becomes negative. The system evolves like that until $U_v(N_d, N_v)$ also reaches 0 or $N_d = 0$. Since we have assumed that the curves do not cross, $N_d = 0$ first. Then N_v increases until we reach the equilibrium $(N_d^*, N_v^*) = (0, \frac{\mu - q_v}{\lambda_v})$ such that $U_d < 0$ and $U_v = 0$ so that no type of application has an incentive to unilaterally change its mean number of emitting sources.

- If the curve $U_d(N_d, N_v) = 0$ is always above $U_v(N_d, N_v) = 0$ (case (b) of Figure 1), we obtain the symmetric case of the previous one, leading to the equilibrium $(N_d^*, N_v^*) = (\frac{\mu - q_d}{\lambda_d}, 0)$.
- If there is a crossing point (N_d^*, N_v^*) for the curves (case (c) of Figure 1), this crossing point is unique from Lemma 1. Note that $U_v(0, N_v) = 0$ for $N_v = \frac{\mu - q_v}{\lambda_v}$, $U_d(N_d, 0) = 0$ for $N_d = \frac{\mu - q_d}{\lambda_d}$ and that $U_d(0, \frac{\mu - q_v}{\lambda_v}) > 0$ and $U_v(\frac{\mu - q_d}{\lambda_d}, 0) > 0$.

  Again, the mean number of emitting sessions will (continuously) increase until one curve $U_d = 0$ or $U_v = 0$ is reached. Denote by $(\tilde{N}_d, \tilde{N}_v)$ the mean numbers of sources when such a point is reached. We can have two different cases:

  - If the first utility function to become null is U_d, then we have $U_d(\tilde{N}_d, \tilde{N}_v) = 0$ and $U_v(\tilde{N}_d, \tilde{N}_v) > 0$ so that N_v will increase, implying that N_d will decrease. Since $\tilde{N}_d > N_d^*$ and $\tilde{N}_v < N_v^*$, it will evolve this way until we eventually reach the point (N_d^*, N_v^*) where $U_d(N_d^*, N_v^*) = 0$ and $U_v(N_d^*, N_v^*) = 0$.
  - If the first utility function to become null is U_v, we can exchange the roles of data and voice, and we arrive to the same conclusion. □

3 Proof of Proposition 2

In order to prove this proposition, we first prove that the condition is sufficient and then that it is necessary.

- To prove that the condition expressed by (5), (6) and (7) is sufficient, assume that q_d and q_v satisfy those relations.

  After some calculations, (6) can be rewritten as

$$\mu\gamma\frac{q_v}{q_d} - (1-\gamma)\mu < \gamma q_d - (1-\gamma)q_v, \tag{8}$$

  and (7) can be rewritten as

$$\mu\gamma - \mu(1-\gamma)\frac{q_d}{q_v} > \gamma q_d - (1-\gamma)q_v. \tag{9}$$

  Indeed, from inequality (7),

$$\begin{aligned} q_d &< \frac{\mu\gamma q_v + (1-\gamma)q_v^2}{\gamma q_v + \mu(1-\gamma)}, \\ \frac{q_d}{q_v} &< \frac{\mu\gamma + (1-\gamma)q_v}{\gamma q_v + \mu(1-\gamma)}, \\ \gamma q_d - (1-\gamma)q_v &< \mu\gamma - \mu(1-\gamma)\frac{q_d}{q_v}. \end{aligned} \tag{10}$$

Also, from inequality (6),

$$\begin{aligned}\sqrt{\mu q_v + (\frac{\mu - q_v}{2}\frac{1-\gamma}{\gamma})^2} - \frac{\mu - q_v}{2}\frac{1-\gamma}{\gamma} &< q_d,\\ \gamma q_d^2 - q_d q_v(1-\gamma) &> \gamma\mu q_v - q_d\mu(1-\gamma), \qquad (11)\\ \gamma q_d - q_v(1-\gamma) &> \gamma\mu\frac{q_v}{q_d} - \mu(1-\gamma).\end{aligned}$$

From (9), we obtain $q_v(\gamma q_d - (1-\gamma)q_v) < \mu\gamma q_v - (1-\gamma)\mu q_d$, and from (8) $\mu\gamma q_v - (1-\gamma)\mu q_d < q_d(\gamma q_d - (1-\gamma)q_v)$.

Our goal is to show that we are in a non-trivial equilibrium, meaning that the (unique) solution of the system $U_d(N_d, N_v) = 0$ and $U_v(N_d, N_v) = 0$ is such that $N_d > 0$ and $N_v > 0$. ¿From the proof of Lemma 1, the solution of the system verifies $\lambda_d N_d = \frac{\mu q_d}{\gamma q_d - (1-\gamma)q_v} - \frac{q_v q_d}{\gamma q_v - (1-\gamma)q_d}$, and $\lambda_v N_v = -\frac{\mu q_v}{\gamma q_d - (1-\gamma)q_v} + \frac{q_v q_d}{\gamma q_v - (1-\gamma)q_d}$. To prove that those values are strictly positive, consider the two complementary cases.

- If $\gamma q_d - (1-\gamma)q_v > 0$, then we have $\gamma q_v - (1-\gamma)q_d > 0$ by a simple manipulation. We deduce from Equation (9) that $\mu > q_v\frac{\gamma q_d - (1-\gamma)q_v}{\gamma q_v - (1-\gamma)q_d}$, which gives that $\lambda_d N_d > 0$. Moreover, we have from Equation (8) that $\mu < q_d\frac{\gamma q_d - (1-\gamma)q_v}{\gamma q_v - (1-\gamma)q_d}$, which gives that $\lambda_v N_v > 0$.
- Similarly, assume $\gamma q_d - (1-\gamma)q_v < 0$. We obtain that $\gamma q_v - (1-\gamma)q_d < 0$ by a simple manipulation. Then $\mu > q_v\frac{\gamma q_d - (1-\gamma)q_v}{\gamma q_v - (1-\gamma)q_d}$, from (9), which gives that $\lambda_d N_d > 0$. Moreover, (8) also gives $\mu < q_d\frac{\gamma q_d - (1-\gamma)q_v}{\gamma q_v - (1-\gamma)q_d}$, from which $\lambda_v N_v > 0$.

The sufficient condition is then proved.

- Second, we prove that the condition is necessary, i.e. that assuming the Nash equilibrium non-trivial, Relations (5), (6) and (7) are verified. N_d^* and N_v^* satisfy $\lambda_d N_d^* = \frac{\mu q_d}{\gamma q_d - (1-\gamma)q_v} - \frac{q_v q_d}{\gamma q_v - (1-\gamma)q_d}$, and $\lambda_v N_v^* = -\frac{\mu q_v}{\gamma q_d - (1-\gamma)q_v} + \frac{q_v q_d}{\gamma q_v - (1-\gamma)q_d}$. As both are non-negative, we obtain: $\frac{q_v}{\gamma q_v - (1-\gamma)q_d} < \frac{\mu}{\gamma q_v - (1-\gamma)q_d}$, and $\frac{\mu}{\gamma q_v - (1-\gamma)q_d} < \frac{q_d}{\gamma q_v - (1-\gamma)q_d}$, which is equivalent to: $\mu\gamma\frac{q_v}{q_d} - (1-\gamma)\mu < \gamma q_d - (1-\gamma)q_v$, and $< \mu\gamma - \mu(1-\gamma)\frac{q_d}{q_v}$. From (10), condition (7) can be rewritten as: $\frac{\mu\gamma + (1-\gamma)q_d}{\gamma q_d + \mu(1-\gamma)} < \frac{q_d}{q_v}$, and from (11), condition (6) becomes $\frac{q_d}{q_v} < \frac{\mu\gamma + (1-\gamma)q_v}{\gamma q_v + \mu(1-\gamma)}$. These inequalities give the two relations between q_d and q_v: $q_d < \frac{\mu\gamma q_v + (1-\gamma)q_v^2}{\gamma q_v + \mu(1-\gamma)}$, and $q_d > \sqrt{\mu q_v + (\frac{\mu - q_v}{2}\frac{1-\gamma}{\gamma})^2} - \frac{\mu - q_v}{2}\frac{1-\gamma}{\gamma}$ by solving the second order polynomial.

  Moreover, $q_v > \mu$ is not possible since $N_v^* < \frac{\mu - q_v}{\lambda_v} < 0$. □

Joint Sensor Selection and Data Routing in Sensor Networks

Ozgur Ercetin, Ozgur Gurbuz, Kerem Bulbul, and Aylin Aksu

Faculty of Engineering, Sabanci University, Istanbul 34956, Turkey
{oercetin@, ogurbuz@, bulbul@, aylinaksu@su.}sabanciuniv.edu

Abstract. We propose a new joint sensor selection and routing algorithm, which selects a set of sensor nodes (sensing nodes) in a sensor network to take measurements, and determines a set of paths connecting the sensing nodes to the sink node. Our objective is to maximize the network lifetime, while satisfying the data precision required by the user. We first develop a multi-objective optimization model for this problem and design the near-optimal OPT-RE algorithm based on this model for network lifetime maximization. Next, we design a low complexity heuristic called SP-RE. SP-RE first labels the links between the nodes with a metric which trades off the residual energies of the transmitting and receiving nodes with the required transmission and reception energy. Then, SP-RE calculates the shortest paths from all nodes to the sink, and identifies the node which is closest to the sink as a *sensing* node. This process is repeated until the required data precision is satisfied. We demonstrate by simulations that SP-RE and OPT-RE can increase the network lifetime several orders of magnitude compared to naive approaches.

1 Introduction

Sensor networks consist of hundreds or thousands of low cost nodes that come with wireless communication and certain processing and storage capabilities in addition to sensing capabilities. One of the most important applications foreseen for sensor networks is environmental monitoring, where the sensor nodes are embedded into a physical environment to monitor and collect data about the ambient conditions. The nodes are deployed in the sensor field and can be thought of as distributed streaming data sources supporting various (monitoring) applications running at the base station. The data is collected from sensor nodes via wireless multi-hop communication. Sensor nodes usually have limited battery capacities and it is impractical to replace the batteries of thousands of sensor nodes after they are deployed in the field. Thus, a key challenge in ad-hoc, data-gathering wireless sensor networks is prolonging the lifetime of the network.

In this paper, we aim to design algorithms which jointly select a set of *sensing* nodes in a sensor network, and the paths from each of these nodes to the sink in an energy-efficient way. The nodes in the sensor network cannot be selected randomly: First, the required data precision should be considered; and second, the residual energy of the nodes should be taken into account. As an illustrative

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 828–839, 2005.

example, consider a simple application in which sensor nodes are used to monitor ambient conditions such as temperature, humidity, light, etc., in a physical area. While monitoring the conditions in this area, one extreme case is to use all the sensor nodes and get almost a complete picture of the conditions. This option leads to maximum possible precision; however, using all nodes in the network leads to very high energy consumption and a short network lifetime. Another extreme solution is to use a single sensor node to monitor the whole area regardless of the area size. This results in a very low resolution picture of the ambient conditions in the area, but it is clear that for this case the energy consumption would be very low. These two solutions are extreme cases and neither of them will be useful in real implementations. In a real implementation, a user or monitoring application will select a certain measurement precision level required by the application. This precision level determines how close the measurements should be taken over the field, e.g., every D square meters. Then, if the sensor network is densely deployed, an important question is to select a node within every such D square meter area for sensing and also to determine an appropriate route from this sensing node to the monitoring station so that the network lifetime is maximized. We investigate the answer to this question in the following sections.

Our work is related to the efforts on topology control and in-network query processing in wireless ad hoc and sensor networks [2], [3], [4], and [5]. Previously in the literature, topology control is investigated with the objective of providing end-to-end traffic, a connected and power/energy efficient path. Our work brings together application-specific requirements (in the form of measurement precision), topology control and routing. The closest model to ours is described in [3]; however, in [3] the objective is to detect the occurrence of an event in an energy-efficient manner rather than to continuously collect the ambient conditions of the area. More recently, Boulis et al. [1] and Kalpakis et al. [6], considered the energy-accuracy trade off in sensor networks when the network is continuously monitored. Boulis et al., investigated the periodic aggregation and estimation problem when the observed data is time correlated. Kalpakis et al., discussed a query-based operation and suggested storing summary data structures at the monitoring station rather than collecting similar data again for each new query. This stored aggregate value is used to answer queries at the monitoring station approximately to improve network lifetime. In our paper, we do not consider in-network processing, but we discuss the optimal selection of sensor nodes and routing of the sensor data given a required measurement precision.

The paper is organized as follows: In section 2, the system model is presented. In section 3, the lifetime maximization problem is formulated as a multi-objective mixed integer program, and a mathematical programming based heuristic for this problem is proposed. In section 4, a low complexity joint node and path selection algorithm is developed. In section 5, results of our performance analysis are presented that include a comparison of our two heuristics along with detailed simulations that benchmark our heuristics against traditional algorithms from the literature. Section 6 involves our conclusions and further research directions.

2 System Model

The sensor network is used to collect information from the sensors distributed in a field that is required for supporting various (monitoring) applications. Each node in the sensor network has the same sensing capability. The nodes can be deployed in this field randomly or according to a predefined topology. If the nodes are deployed randomly, we assume that the nodes can determine their respective locations by using methods such as those proposed in [7]. For ease of instruction, we discuss our model for the case when sensor nodes are deployed in 1-dimension, i.e., nodes are deployed on a line emanating from the monitoring station. It is easy to extend this model to 2-dimensions.

We consider a query-based operation: A user injects queries to the network, which includes a precision requirement in the form of "measurement every D meters". Upon receipt of this query, the network selects the set of sensing nodes among a number of candidate nodes (nodes that are within ϵ distance to the required measurement points). Note that if the total number of sensor nodes in the network and their locations are known, then defining the precision by the measurement interval is equivalent to defining the precision by the number of sensing nodes and how far off they can be located from each required measurement point. For example, if there are 100 nodes randomly distributed over a line of 100 meters, then asking for a measurement precision of every 10 meters is equivalent to selecting 10 sensing nodes that are as equidistant to each other as possible.

Once the sensing nodes are selected, the sensor data has to be forwarded to the monitoring station. In general, the message follows a multi-hop path from a sensing node to the destination. The path is defined by the set of relay nodes and their transmission powers. We assume that the transmission power of each node is uniquely determined by the distance between the transmitting node and the next node on the path receiving the message.

3 Lifetime Maximization Problem

The network lifetime is defined as the duration over which the network can respond to the measurement queries. In general, every query may require a different precision which we do not know in advance. In other words, the generation of queries over time is a random process, and we cannot maximize the network lifetime unless we know the distribution of this random process. Therefore, in order to prolong the network lifetime we propose a slightly different approach motivated by the following two observations: first, when data is routed from the sensing nodes to the monitoring station, it makes sense to avoid using nodes with small residual energy so that no node dies too early. Second, we would like to minimize the total energy consumption from sensing, transmission and receiving when responding to a query. Thus, by not depleting the energy of any individual node too quickly and minimizing the total energy consumption, we intend to increase the network lifetime. Below, we formulate these two goals in a multi-objective optimization framework and then explain briefly how we solve this optimization model.

Assume that a given query Q requires that K sensing nodes are chosen out of a total of N nodes, and data from these sensing nodes are routed to the monitoring station. The parameter K is the required precision of query Q and is related to the interval between successive measurements as discussed in Section 2. Note that the nodes are deployed randomly in 1-dimension within a measurement interval of I units, and let x_i denote the location of node i, $i = 1, \ldots, N$. Without loss of generality, the monitoring station is denoted by the index $i = 0$ and located at $x_0 = 0$. The nodes are numbered in non-decreasing order of their respective distances to the monitoring station. The main sources of energy consumption are the sensing, transmission and receiving operations. Let E_i, $i = 1, \ldots, N$, be the initial energy level of node i before Q is processed, and let ξ and e_r be the constant amount of energy required for a single sensing and receiving operation by any node, respectively. The transmission energy depends on the distance between two nodes, and e_{ij}, $i = 1, \ldots N$, $j = 0, \ldots, N$, denotes the amount of energy consumed by node i when sending a unit size packet to node j. The messages generated by the K sensing nodes are routed to the monitoring station over multi-hop paths where the binary variable f_{ij}^k, $k = 1, \ldots, K$, $i = 1, \ldots, N$, $j = 0, \ldots, N$, denotes the number of packets generated by a sensing node k and transmitted from node i to node j.

Ideally, we would like to select K sensing nodes in the network such that they are approximately equidistant from each other while sensing data at the required precision of query Q. Note that if the nodes are randomly deployed, we may not always find a sensing node that is located exactly I/K units away from the previous sensing node. Moreover, as discussed in the previous section, it may not be optimal to choose the same sensing nodes for every query over time. Therefore, we allow the distance between two successive sensing nodes to change between $I/K * (1 - \epsilon)$ and $I/K * (1 + \epsilon)$. In addition, we require that the first and last sensing nodes are located within $I/K * (1 + \epsilon)$ units of the lower and upper bounds of the measurement interval, respectively. (In our computational experiments we set $\epsilon = 0.1$.) In order to impose these constraints on the relative locations of the sensing nodes, we construct the sets $S_L = \{i \,|\, x_i \leq \frac{I}{K} * (1 + \epsilon)\}$, $S_i = \{j \,|\, x_i + \frac{I}{K} * (1 - \epsilon) \leq x_j \leq x_i + \frac{I}{K} * (1 + \epsilon)\}$, $i = 1, \ldots, N$, and $S_U = \{i \,|\, x_i \geq I - \frac{I}{K} * (1 + \epsilon)\}$. We also define the binary variables s_i^k, $i = 1, \ldots, N$, $k = 1, \ldots, K$, where $s_i^k = 1$ if node i is the k^{th} sensing node, and $s_i^k = 0$ otherwise. Finally, let L_i, $i = 1, \ldots N$, be a variable that denotes the residual energy level of node i after Q is processed and $W = \min_i L_i$ be the final, i.e., residual, energy of the node with the smallest final energy in the network. Then, we formulate the multi-objective residual energy optimization problem OPT-RE as given in (1)-(11) below.

$$\max\{z_1 = W,\ z_2 = \sum_{i=1}^{N} L_i\} \tag{1}$$

$$L_i \geq W \quad \forall i \tag{2}$$

$$E_i - \xi \sum_{k=1}^{K} s_i^k - \sum_{k=1}^{K}\sum_{j=0}^{N} e_{ij} f_{ij}^k - e_r \sum_{k=1}^{K}\sum_{j=1}^{N} f_{ji}^k - L_i = 0 \quad \forall i \tag{3}$$

$$\sum_{j=0}^{N} f_{ij}^k - \sum_{j=1}^{N} f_{ji}^k - s_i^k = 0 \quad \forall i \neq 0, \forall k \tag{4}$$

$$-\sum_{i=1}^{N} f_{i0}^k = -1 \quad \forall k \tag{5}$$

$$\sum_{i \in S_L} s_i^1 = 1 \tag{6}$$

$$s_i^k \leq \sum_{j \in S_i} s_j^{k+1} \quad \forall i,\ \forall k \neq K \tag{7}$$

$$\sum_{i \in S_U} s_i^K = 1 \tag{8}$$

$$L_i \geq 0 \quad \forall i \tag{9}$$

$$f_{ij}^k \in \{0,\ 1\} \quad \forall i, j, k \tag{10}$$

$$s_i^k \in \{0,\ 1\} \quad \forall i, k \tag{11}$$

The multi-objective function (1) reflects our two goals after the query Q is processed: maximize the residual energy of the node with the minimum residual energy (objective z_1) while also maximizing the total residual energy of the network (objective z_2). There exists a trade-off between these two objectives, and we will further elaborate on this issue below. The constraints (3) relate the initial energy levels E_i to the final energy levels L_i. The constraints (4) represent the flow conservation constraints for nodes $i = 1, \ldots, N$ assuming that a single packet is generated by a sensing node for each query. At a sensing node i, i.e., when there exists some k such that $s_i^k = 1$, the net flow out of the node is equal to 1 which corresponds to the packet generated at this node. Otherwise, when node i is a relay node, i.e., when $s_{ik} = 0\ \forall k$, the net flow at node i is zero. The constraints (5) ensure that exactly one packet arrives at the monitoring station from each sensing node. Constraints (6)-(8) select K sensing nodes in the network such that they are approximately equidistant from each other while sensing data at the required precision of query Q.

In order to solve OPT-RE, we need to specify the exact relationship between the two objectives z_1 and z_2 in (1). The following case illustrates why we need a multi-objective model for this network lifetime maximization problem and proposes a way of solving OPT-RE. Assume that we need to process a given query Q for m times and our only objective is to maximize the total residual energy after Q_m is processed. Clearly, if we determine the optimal sensing nodes and the optimal routes to the monitoring station such that the total energy consumption for processing the first query Q_1 is minimized, then the optimal sensing nodes and routes for processing the queries $Q_2, \ldots, Q_m$ are exactly the same unless a node dies while one of these queries is being processed. In other words, maximizing z_2 subject to the constraints (3)-(11) yields a solution that uses the same nodes over and over again until some nodes exhaust their total energy.

Obviously, this approach does not lead to an increased network lifetime, and we must make sure that we do not overuse a subset of the nodes in the network over time. Thus, we implement the following multi-objective approach: First, we maximize z_1 subject to the constraints (2)-(11) given a query Q_t and the initial energy levels $E_i^t,\ i = 1, \ldots, n$, and obtain the optimal minimum energy W^* in the network. Then, for the same query Q_t and the same initial energy levels $E_i^t,\ i = 1, \ldots, n$, we maximize z_2 subject to the constraints (2)-(11) except that we replace the right hand side of (2) with W^*. In other words, we maximize the total residual energy subject to the additional constraint that no node can have a residual energy less than W^* after Q_t is processed. The optimal solution of this second single-objective optimization problem yields us the optimal locations of the sensing nodes and the optimal routes to the monitoring station for query Q_t along with the final energy levels L_i^t of the nodes in the network. Then, we set $E_i^{t+1} = L_i^t,\ i = 1, \ldots, n$ and solve this multi-objective optimization problem for the next query Q_{t+1}.

The approach described above requires us two solve two mixed integer programs for each query processed, and thus is not practical. Furthermore, this is still a myopic approach for maximizing the network lifetime because when we select the sensing nodes and the optimal routes for a query Q_t, we completely ignore the queries to be processed in the future. Nevertheless, we expect this mathematical programming based heuristic to provide a good solution in general. In the next section, we propose another heuristic that is easier to implement and computationally much faster. In Section 5, we compare these two heuristics for practical size networks and demonstrate the effectiveness of both algorithms.

4 Joint Node Selection and Routing Algorithm

Due to the computational complexity of OPT-RE, we propose a second algorithm with a simpler heuristic, which determines the set of sensing nodes as well as the paths over which the sensor data is carried to the base station, while solving the lifetime maximization problem. Our new heuristic considers a new link cost function c_{ij} for link (i, j) with four parameters: the energy expenditure for transmitting a unit size packet over the link, e_{ij}; the initial energies of nodes i and j, E_i, and E_j; and the energy expenditure for the reception of a unit size packet, e_r. Note that the energy expenditure due to radio reception is usually considered as a constant. A good sensing node and path choice should consume as little total amount of energy as possible and should avoid nodes with small residual energy. Therefore, the link cost function should be such that when the transmitting and receiving nodes have plenty of residual energy, the energy expenditure terms are emphasized, and if the residual energies of the transmitting and receiving nodes become small, the residual energy terms dominate the link cost. Thus, we propose a link cost of the form

$$c_{ij} = e_{ij}/E_i + e_r/E_j. \tag{12}$$

The path cost is computed by the summation of the link costs on the path, and the shortest path can be found by using any of the existing shortest path (SP) algorithms (e.g., Dijkstra's algorithm). For this reason, this new heuristic approach is named as SP-RE.

The general description of the node selection and routing algorithm for a given query is given in Figure 1.

```
Graph $G = (V, A)$ is given where $V$ is the set of sensor nodes and $(l, m) \in A$ is
the edge connecting nodes $l$ and $m$.
  1. Initially set $k = 1$, and select $K$ sensing nodes. While $k \leq K$:
     – Label each link $(l, m)$ in $A$, $c_{lm} = e_{lm}/E_l + e_r/E_m$.
     – Determine the shortest paths from each node in $V$ to the sink node.
       Order the costs of the shortest paths in non-decreasing order, where
       $SP_i = \{\text{sink}, g^i_1, g^i_2, \ldots, g^i_{|SP_i|-2}, n_i\}$ is the $i$th lowest cost shortest path,
       and let $n_i$ denote the sensing node corresponding to $SP_i$.
     – $i = 1$: while $s^k_{n_i} = 1$ does not satisfy constraints (6)-(8), $i = i + 1$.
     – Select $n_i$ as the sensing node, set $s^k_{n_i} = 1$, and update the link costs of
       all nodes on the path $SP_i$ with new residual energy values where $E'_{n_i} =$
       $E_{n_i} - \xi - e_{n_i, g^i_{|SP_i|-2}}$ for the sensing node and $E'_{g^i_t} = E_{g^i_t} - e_{g^i_t, g^i_{t-1}} - e_r$
       for the relay nodes.
  2. $k = k + 1$ and go to step 1.
```

Fig. 1. Pseudocode for SP-RE

In the first iteration, the monitoring station determines the shortest cost paths to every node in the network. Then, among all nodes in the network the node with the shortest cost path to the destination is selected to sense and the sensor data flows over its shortest cost path to the destination. The link metrics are updated according to this decision. In the next iteration, the monitoring station again determines the shortest cost paths to every node in the network, but this time with the updated link metrics. If the location of the node with the shortest cost path to the monitoring station does not satisfy the constraints (6)-(8), then we examine the node with the next shortest path to the destination and continue until we determine the next sensing node that satisfies the conditions (6)-(8) on the relative locations of the sensing nodes. We continue to iterate until the requested number of sensor nodes are selected.

5 Performance Analysis

In this section, we first present the comparison of the performances of OPT-RE and SP-RE algorithms. Figure 2 shows the variation of minimum energy (E_{min})

and average energy (E_{avg}) versus time measured in a network consisting of 200 nodes randomly deployed over a line of 100 meters. Queries are sent every second and the measurements are taken every 50 meters, i.e., $K = 2$, with 2 sensing nodes. Note that the locations of the nodes are static, and the same network topology is used for both OPT-RE and SP-RE. The initial battery capacities of the nodes are randomly selected between 0.6 and 1.4 joules. Again, the same battery capacities are used for both algorithms. The node energies are decremented according to the Crossbow specifications [9]. Only the energies consumed due to the sensing and radio operations are included in the performance evaluations.

The solution for OPT-RE is obtained by using CPLEX and OPLStudio [10], which determines the sensing nodes, routes to the base station, the energy expenditures and the residual energy levels of all nodes. The solution is updated every 3600 queries (which corresponds to updating the sensor and path selections every hour), and E_{min} and E_{avg} are determined until the first node exhausts its energy. The heuristic SP-RE runs Dijkstra's algorithm with the link costs given in (12) for computing the shortest paths, and the solution is updated every hour.

In Figure 2, we observe that the network lifetime achieved by SP-RE is quite close to that attained by OPT-RE; the difference is less than 9%. Thus, SP-RE appears as a promising joint sensor selection and routing algorithm that performs similarly to more complicated optimization-based algorithms. When we examine Figure 2 more carefully, we observe certain characteristics of the two algorithms. With SP-RE, the minimum node energy drops quickly after several hours of operation in contrast to OPT-RE, where the algorithm keeps the minimum node energy constant for a long time. Meanwhile, the average node energy with OPT-RE drops below the average node energy attained by SP-RE. The reason is clear; unlike SP-RE, the objective of OPT-RE is to maximize the minimum residual node energy. Thus, OPT-RE tries to choose nodes with energies above the minimum energy level for sensing and routing before re-using nodes with minimum energy. In doing so, the average energy in the network drops quickly, since most of the nodes in the network tend to have similar residual energies.

Unfortunately, the required running time with OPT-RE prohibits us to perform the rest of the experiments with OPT-RE. Thus, in the following, we compare the network lifetime achieved by SP-RE with other non-optimization-based heuristic approaches; TinyDB and TinyDB-RE. We believe that this is acceptable considering Figure 2, where it is shown that SP-RE closely matches the network lifetime achieved by OPT-RE. The first method TinyDB [8] is the standard query and routing algorithm used in Crossbow motes. In TinyDB, each node transmits at a predetermined, fixed constant power level, and the nodes separately determine the shortest route to the base station. In this case, sensor selection is done only based on the physical proximity to the desired measurement point. The second algorithm, TinyDB-RE computes the routes to the base station in the same way as TinyDB; however, the node with the highest residual energy within the required proximity to the measurement point is selected for sensing.

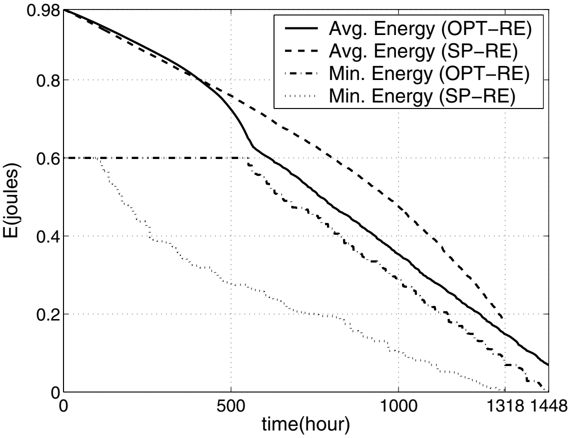


Fig. 2. Residual energy of nodes with OPT-RE and SP-RE

Figure 3 shows the variation of average and minimum energy levels measured over the lifetime of the network using SP-RE, TinyDB and TinyDB-RE. There are 100 nodes distributed over a line of 100 meters, and queries require a measurement every 50 meters. All nodes have the same initial energy of 1.4 joules. The query is repeated every 1 second, whereas the sensing node and path selection is updated every hour (every 3600 queries). As depicted in Figure 3, the lifetime of the network using SP-RE is around 330 hours. The simulations indicate that the other two methods perform poorly as compared to SP-RE, with an order of magnitude decrease in the lifetime. More specifically, when TinyDB is used, the lifetime of the network is measured as 7 hours. TinyDB-RE improves the lifetime to around 15 hours, with intelligent sensor selection. Nevertheless, since both algorithms apply constant transmission power, the algorithm SP-RE significantly enhances performance with smart selection of power levels and routing.

Next, we consider several scenarios with different node densities and query precisions, and compare the lifetimes achieved by three methods. First, we look into how the network lifetime is influenced by the increasing node density. Figure 4 depicts the measured lifetime with the three algorithms for values of N ranging between 10 and 600 where the nodes are placed randomly over a line of 100 meters. The sampling interval is set to 20 meters, i.e., $K = 5$. From Figure 4, we can infer that the lifetime with TinyDB or TinyDB-RE is not affected by the increasing node density, resulting in a lifetime of approximately 20 hours. Note that TinyDB performs the sensor node selections only once based on the physical proximity to the measurement points, and the routes of packets from those sensing nodes to the base station remain unchanged. When the algorithm TinyDB-RE is used, the sensing nodes can be changed according to the residual energy levels. However, the routing of sensor data is still not performed based on residual energies. Finally, when SP-RE is employed, both the sensing node and transmission power selections are performed dynamically, and routes to the

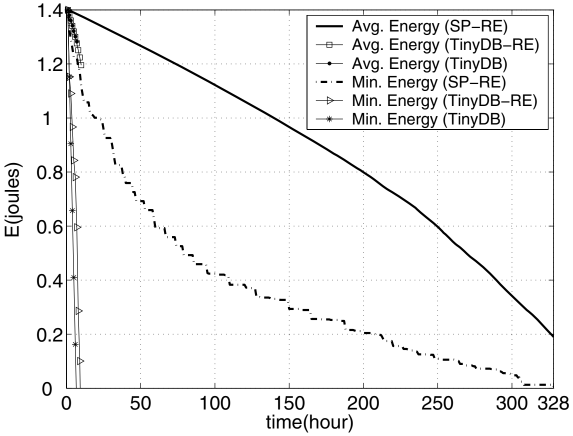


Fig. 3. Lifetime comparison of SP-RE with TinyDB and TinyDB-RE

base station are updated adaptively. Increasing the node density provides alternate sensing nodes and paths to the sink, and thus the lifetime is increased significantly. In our experiments, the measured lifetime attained by the algorithm SP-RE is about 2000 hours which is much higher those achieved by other methods.

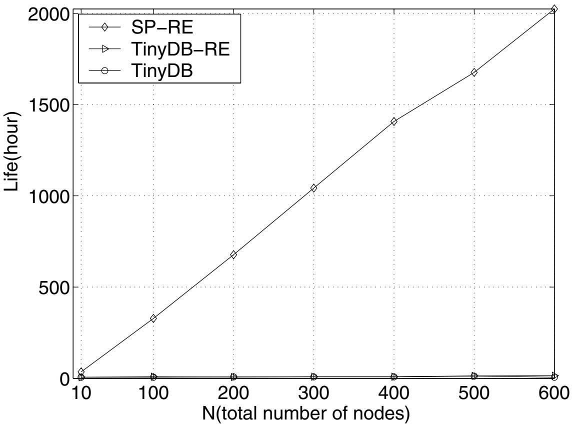


Fig. 4. Lifetime of SP-RE, TinyDB and TinyDB-RE with respect to varying node density (sampling interval = 20 meters)

In Figure 5, we evaluate the variation of lifetime with respect to the sampling interval for a network with 500 nodes uniformly distributed in an interval of 100 meters, and compare the behavior of the three algorithms. Our experiments show that the lifetime of the network remains the same when TinyDB is used, independent of the network and the sizes of the sampling intervals. The performance of TinyDB-RE is improved with the increased sampling interval to about the twice of the lifetime achieved by TinyDB. This is due to the fact that TinyDB-RE selects the sensor nodes based on residual energies, and increasing the sampling interval increases the choices for sensing nodes. Meanwhile, the lifetime attained by SP-RE increases significantly with increasing sampling interval. This behavior is especially apparent in dense networks such as the one considered in this experiment.

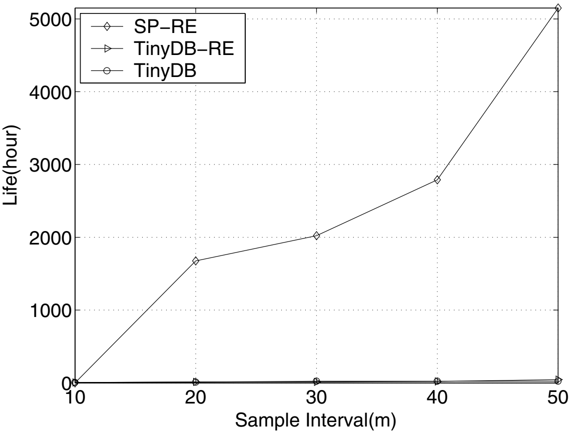


Fig. 5. Lifetime of SP-RE, TinyDB and TinyDB-RE with respect to varying sampling intervals

The computational experiments show that SP-RE is definitely a promising algorithm as compared to both optimization-based and naive methods for enhancing the lifetime of sensor networks, due to the incorporated features such as intelligent sensor selection, transmission power control and data routing. These experiments, although preliminary, provide us some insight about the extent of the advantages that SP-RE can provide. Our results show that these advantages become more significant as the network density is increased, which is quite valuable for environment monitoring applications.

6 Conclusions and Future Work

In this paper, the trade off between measurement precision and energy efficiency is explored through the lifetime of a sensor network. The goal is to maximize the network lifetime by determining the set of sensing nodes and the radio transmission powers for all participating nodes, i.e., network topology, while satisfying the required measurement precision. A mathematical programming based solution and a low complexity heuristic algorithm are proposed for the energy-precision optimization problem. Our results indicate that with only 10% of deviation from the requested exact measurement intervals, the improvement in network lifetime using our algorithms is significant compared to the algorithms currently available in the literature.

However, in this paper we solved only a part of the energy-efficient monitoring problem: We assumed that a user is interested in equally spaced measurements. In a more general setting, a user may be interested in non-uniform measurements to depict a closer approximation of the current conditions in the field. This problem has the promise to bring together signal processing and networking fields, since the current conditions in the field can be viewed as a 2-D image that is being encoded by the nodes in sensor network in an efficient way. One of our future goals is to further investigate this problem.

Another future extension of our work is to consider the energy consumption due to MAC; collisions and retransmissions in a realistic wireless network environment may cause increased consumption of energy.

References

1. A. Boulis, S. Ganeriwal, and M.B. Srivastava, "Aggregation in Sensor Networks: An Energy-Accuracy Trade-off. Sensor Network Protocols and Applications," SNPA 2003, 11 May 2003.
2. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proc. of ACM International Workshop on Wireless Sensor Networks and Applications, WSNA '02*, Atlanta, GA, September 2002.
3. S. N. Simic and S. Sastry, "Distributed Environmental Monitoring Using Random Sensor Networks," *Proc. of IPSN'03*, Palo Alto, CA.
4. N. Li and J. C. Hou, "Topology Control in Heterogeneous Wireless Networks: Problems and Solutions," Proc. IEEE INFOCOM 2004, March 2004
5. S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar,"Power Control in Ad-Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW Protocol," in Proceedings of European Wireless 2002.
6. K. Kalpakis, V. Puttagunta and Parag Namjoshi,"Accuracy Vs Lifetime: Linear Sketches for Appoximate Aggregate Range Queries in Sensor Networks," Department of Computer Science Technical Report, TR CS-04-04, University of Maryland, Baltimore County, 2004.
7. E. D. Kaplan, Ed., "Understanding GPS Principles and Applications.Norwood," MA: Artech House, 1996.
8. S. Madden, J. Hellerstein, W. Hong, "TinyDB: In-Network Query Processing in TinyOS," Sep 2003.
9. http://www.xbow.com, 2004.
10. http://www.ilog.com, 2004.

Peer Collaboration in Wireless Ad Hoc Networks

Lin Cai[1], Jianping Pan[2], Xuemin Shen[1], and Jon W. Mark[1]

[1] University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
[2] NTT MCL, Palo Alto, California 94306, USA

Abstract. Voluntary peer collaboration is often assumed in media access, route discovery, packet forwarding, and upper-layer protocols for wireless ad hoc networks. This assumption is seriously challenged when some peers are autonomous, selfish, or malicious in large-scale, heterogeneous networks. In this paper, based on the latest advances in identity-based cryptography, we design a lightweight and cheat-resistant micropayment scheme to stimulate and compensate collaborative peers that sacrifice their resources to relay packets for other peers. We also demonstrate that when security and collaboration measures are properly enforced, profitable collaboration is a preferable strategy for all peers.

1 Introduction

Wireless ad hoc networks are self-organized systems without relying on any preexisting, fixed communication infrastructures, so individual peers have to assist communications that are vital for other peers. Wireless ad hoc networks are especially attractive when infrastructures are too expensive to build, or too vulnerable to maintain [1,2], and have attracted much attention in recent years [3,4]. *Voluntary collaboration* is often assumed among involved peers, which is acceptable when all peers are genuine, collaborative, and under the control of a single authority. As indicated in [5,6,7,8,9,10,11,12], the validity of this assumption is challenged when some peers are autonomous, selfish, or malicious. For example, if battery-powered peers relay packets for other peers, they are one-step closer to running out of their energy, which is undesirable from a selfish standpoint, since they may have insufficient energy to send their own packets later.

In this paper, we are interested in secure collaboration of *selfish* peers in energy-constrained wireless ad hoc networks. In our setting, a peer (e.g. a user carrying a battery-powered laptop computer with wireless LAN interfaces) joins a group of other peers. These peers may or may not have preestablished trustworthiness (e.g. in a public recreation park), or share any common goals (e.g. accessing the Internet or swapping files). A peer may raise the output power of its transmitter to communicate with intended peers directly; however, its capability to do so in practice is always limited by hardware design, and such a strategy may not be preferred by other peers (due to higher interference) or even by itself (due to higher energy consumption). Hence, collaborations among neighboring peers (e.g. relaying) are essential in wireless ad hoc networks.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 840–852, 2005.

The desire to collaborate in wireless ad hoc networks faces many new challenges. First, peers have to be assured that they indeed exchange information with intended peers, even when they no longer communicate with each other directly. Second, as packets are relayed by peers without preestablished trustworthiness, peers have to be assured that the confidentiality, integrity, and authenticity of exchanged information are not compromised. Third, selfish peers always want to take advantage of other peers, but hesitate to help others if their resources are sacrificed, so certain measures are required to stimulate and compensate favorable collaborations. Finally, the entire system should benefit from secure collaboration among selfish peers, and resist against malfunctioning or malicious peers; otherwise, peers tend to remain selfish.

In contrast to many existing approaches (see Sect. 5 for related work), we apply the latest advances in identity-based cryptography (IBC) [13] to ad hoc networks. IBC is a form of public-key cryptography (PKC). Unlike regular PKC systems, in which the binding between the identity of an entity and its public-key should be certified by certificate authorities (CAs) or stored in central directories, such authorities and directories can be completely eliminated in IBC systems, in which the public-key of an entity can be derived from its identity directly. This property is vitally important for wireless ad hoc networks, where public-key infrastructures (PKIs) or CA hierarchies are generally expensive to build and maintain. IBC is used to facilitate asymmetric encryption/decryption and signature/verification procedures; it can also be used to bootstrap their symmetric counterparts without prearranging pairwise shared secrets among all involved peers. Based on IBC, a lightweight and cheat-resistant micropayment scheme can be devised for ad hoc networks, which stimulates and compensates collaborative peers that sacrifice their resources to help others.

The remainder of this paper is organized as follows. In Sect. 2, we present the model for ad hoc networks, their security requirements, and our IBC-based approaches. In Sect. 3, we design an IBC-based micropayment scheme to stimulate and compensate collaborative peers. Through the performance studies in Sect. 4, we show that profitable collaboration is preferable if properly enforced. Sect. 5 reviews related work, and Sect. 6 concludes this paper.

2 Secure Communications

Network model — As shown in Fig. 1, wireless ad hoc networks are fully-distributed systems of self-organizing peers that wish to exchange information over-the-air but do not rely on any preexisting infrastructures [1,2,3,4]. Mobile peers (e.g. laptop computers, shown as dots, with wireless interfaces) can join or leave such systems (depicted by a large dashed circle, e.g. a recreation park) at any time. Only peers require keying have to pass by an offline authority regularly (e.g. a ticketing booth within a small dotted circle). Without any centralized online authorities, peers can remain stationary or mobile, keep idle (unfilled dots) or active (filled dots), and assist others if they choose to do so.

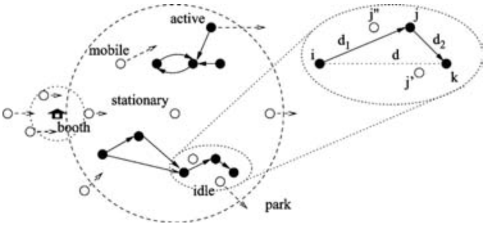


Fig. 1. Relaying in wireless ad hoc networks

Let peer i in Fig. 1 transmit a bulk of data b to another peer k that is d away. i can have two options: 1) i transmits b to k directly, and consumes energy $e_i^t(b,d) = (t_1 + t_2 d^{n(d)})b$, where $2 \leq n(d) \leq 6$ is the path loss exponent, and t_1 and t_2 are the coefficients of distance-independent and related energy consumption. i cannot always do so if $d > D$ and D is its maximum transmission range. 2) When there is a third peer j between i and k, i may save energy by requesting j to relay b to k. Without loss of generality, assume j is d_1 away from i, and d_2 from k. If $d_1 < d$, relaying b through j is preferable for i, while j has to volunteer $e_j^r(b) = r_1 b$ to receive b from i, $e_j^t(b, d_2) = (t_1 + t_2 d_2^{n(d)})b$ to transmit b to k, and $e_j^o(b)$ to cover local expenses. If $e_i^t(b,d) - e_i^t(b,d_1) > e_j^r(b) + e_j^t(b,d_2) + e_j^o(b)$, relaying through j is also preferable for the entire system, since overall it takes less energy to move the same b from i to k. When all peers are voluntary and relaying is favorable, relaying should be mandated. But if j is autonomous, it has no incentive to relay b from i to k; if j is selfish, it will refuse to relay b, since j has to sacrifice its own resources for the benefit of others.

If k is the beneficiary of transferring b directly from i, k should be willing to pay i at least $c_i(b,d) + c_i(b)$ to cover the communication expense occurred at i and the cost for i to obtain b. $c_i(b,d)$ is proportional to $e_i^t(b,d)$ and may be reversely proportional to the remaining energy ϵ_i of i. When relaying is favorable, k finds that it is more cost-effective to retrieve b from j, after j has obtained b from i. To do so, j has to pay i at least $c_j(b) = c_i(b,d_1) + c_i(b)$, and k has to pay j at least $c_j(b,d_2) + c_j(b)$ in advance. If $c_i(b,d) > c_i(b,d_1) + c_j(b,d_2)$, k has enough cost-saving to share with j. For simplicity, we assume j and k share the cost-saving equally; i.e., the net cost-saving at k is $[c_i(b,d) - c_i(b,d_1) - c_j(b,d_2)]/2$, which is also the profit j can make through its relaying.

Security model — Many security threats appear in wireless ad hoc networks [14]. Peers can join or leave at any time without notice, and pairwise trustworthiness among all peers is impractical to build and unrealistic to maintain. Autonomous peers have reasons and excuses to eavesdrop or corrupt relayed data. Malicious peers can impersonate other peers to steal genuine information or inject false information. When relaying is profitable, selfish peers have strong incentives to boost their wealth improperly, by cheating source, destination, or other relaying peers. When there is a certain number of colluding peers, they may even attempt to fool or beat the entire system.

Traditional cryptographic techniques are employed to provide certain security properties in networks with trusted infrastructures. Similar efforts have been attempted in wireless networks: source and destination peers should authenticate to each other before information exchange; also, information should be encrypted by sources to keep confidentiality, and be verified by destinations to preserve integrity. These procedures rely on either certified public-keys in PKC systems, or pairwise prearranged secrets in symmetric cryptography systems. If there are trusted infrastructures (e.g. genuine PKIs or base-stations in cellular systems), such prerequisites can be satisfied accordingly.

However, these techniques do not readily apply to wireless ad hoc networks. First, there are no genuine PKIs or online authorities that can always be involved in communications among any peers. Second, most end-to-end communications in ad hoc networks occur in a hop-by-hop manner, where untrusted third-parties are required to relay packets, so security proprieties should be achieved not only at the end-to-end level, but also at the per-hop level. For example, in Fig. 1, j pays i to obtain b for k; but j's neighbor j' can overhear the communication between i and j, and offers b to k at a lower price. Finally, most existing electronic payment schemes either rely on online, interactive authorities (e.g. banks), or are too heavy (in terms of computation and communication complexity) for wireless ad hoc networks, where energy constraints are the foremost concern.

IBC-based approaches — The concept of IBC was first introduced by Shamir two decades ago [15]. The first efficient and secure IBE scheme (BF-IBE) was given in 2001 by Boneh and Franklin, which employs Weil pairing on elliptic curves [16]; its security is based on the bilinear Diffie-Hellman problem (BDHP), which is considered secure in the random oracle model (ROM).

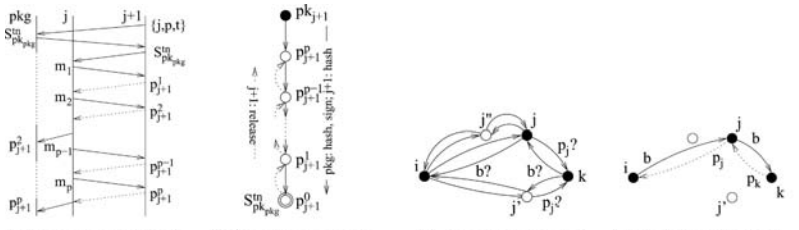
In IBC-based wireless ad hoc networks, each peer proposes its identity (e.g. *a@b.com*, service name, content hash, etc.), which is also its public-key [17]. A private-key generator (PKG, e.g. the ticketing booth in recreation park) verifies identity ownership, appends timestamp for unique identities, and extracts a corresponding private-key from the public system parameters and the master-key only known to PKGs. To reduce the risk of total-exposure with compromised PKGs, and to ease the concern of key escrow with bogus PKGs, the master-key can be distributed in a t-of-n manner to n PKGs with threshold cryptography [18]. Also, hierarchical PKGs allow communications with roaming peers to be protected by their identity and the system parameters of root PKG [19].

When a peer i sends a message m to another peer k, m is encrypted with k's identity id_k and the system parameters; only k can decrypt the encrypted $\hat{m}$ with its private-key pk_k and the system parameters. When k acknowledges m, the receipt is signed with pk_k, and is verifiable by everyone knowing id_k. i knows id_k when communicating with k, and no one else can compromise these procedures without knowing pk_k. IBC also supports authenticated and signed encryption. In addition, IBC can establish a shared-key $sk_{i,k}$ for i and k from their identity id_i and id_k. Using bootstrapped shared-keys, instead of pairwise prearranged secrets, peers can utilize symmetric and more efficient encryption/decryption and message authentication schemes (e.g. HMAC).

Our next step is to stimulate selfish peers to collaborate (i.e. relaying for others), and compensate them if they do so. Here, we focus on a receiver-payer model; other payment models (e.g. sender-payer) can be accommodated by prefixing application-layer payments to our scheme.

3 Collaborative Communications

Hop-by-hop transactions — We first focus on the data transfer and payment scheme between two adjacent peers, j and $j+1$. Assume $j+1$ is willing to pay at least $p = c_j(b) + c_j(b,d)$ to obtain b from j, and j agrees. As shown in Fig. 2(a), to facilitate this transaction with sequence number tn, $j+1$ securely contacts a non-interactive entity (for simplicity, we assume the PKG plays this role) to commit deposited credits of amount p to this transaction. For notational convenience, we assume j, $j+1$, and the PKG have bootstrapped pairwise shared-keys from their identity. The commitment proposal message sent by $j+1$ is $CPPS\{id_{pkg}, tn, id_{j+1}, id_j, p, et\}_{sk_{pkg,j+1}}$, where et indicates the expiry time. As shown in Fig. 2(b), the PKG hashes $j+1$'s private-key pk_{j+1} with j's identity id_j repeatedly for $p+1$ times, i.e. $p_{j+1}^{p-1} = H_{id_j}(tn||p_{j+1}^{p}||et)$, and signs p_{j+1}^{0} with its own private-key pk_{pkg}, i.e. $S_{pk_{pkg}}^{tn} = S_{pk_{pkg}}(tn||p_{j+1}^{0}||et)$, where $S_{pk}(\cdot)$ is the signing procedure with key pk. Only $S_{pk_{pkg}}^{tn}$, instead of the entire hash-chain, is required to be sent back to $j+1$ securely; only p_{j+1}^{0} is kept by the PKG as a record of this transaction. The commitment confirmation message sent by the PKG to $j+1$ is $CCFM\{id_{j+1}, tn, id_{pkg}, id_j, S_{pk_{pkg}}^{tn}, et\}_{sk_{pkg,j+1}}$.



(a) payment schedule (b) payment chain (c) relay negotiation (d) relay commitment

Fig. 2. Per-hop and end-to-end transactions

$j+1$ rebuilds the hash-chain from its private-key pk_{j+1}, and securely reveals $S_{pk_{pkg}}^{tn}$ to j in $HCMT\{id_j, tn, id_{j+1}, S_{pk_{pkg}}^{tn}, et\}_{sk_{j,j+1}}$. The authenticity of $S_{pk_{pkg}}^{tn}$ can be easily verified by j alone with the PKG's identity id_{pkg}. Once j starts to transmit message m_i of b to $j+1$ in $HMSG\{id_{j+1}, tn, id_j, m_i, et\}_{sk_{j,j+1}}$, and after the authenticity of m_i is verified, $j+1$ releases the hash-chain gradually in a reverse order (i.e. p_1, p_2, $\cdots$, p^p) in $HPMT\{id_j, tn, id_{j+1}, p_{j+1}^{i}, et\}_{sk_{j,j+1}}$. j can verify the authenticity of p_{j+1}^{i} alone by keeping a copy of p_{j+1}^{i-1}, the last payment from $j+1$, since $p_{j+1}^{i-1} = H_{id_j}(tn||p_{j+1}^{i}||et)$. The discrepancy of the data transfer and payment between j and $j+1$ is at most one unit. j only keeps the latest (instead of every) payment from $j+1$ as its record.

At any time, j can submit the latest payment from $j+1$ to the PKG in $HCLM\{id_{pkg}, tn, id_j, id_{j+1}, p^i_{j+1}, et\}_{sk_{pkg,j}}$ to claim the actual compensation. The PKG only keeps the latest record submitted by j of the hash-chain committed by $j+1$ as its record. j has to inverse a one-way hash function to claim the payment not received yet (i.e. false claim), which is not feasible.

In our design, $j+1$ cannot deny released payments after it receives m_i from j (except at most one unit discrepancy), since if j has p^i_{j+1}, the PKG knows $j+1$ should have paid j for i times of unit amount. If $j+1$ overspends the hash-chain, it has to reveal its private-key to j, which leads to a more serious consequence for $j+1$ (e.g. j can impersonate $j+1$ to claim $j+1$'s remaining balances). $j+1$ cannot double-spend a single payment without being detected by j, since j always verifies the latest payment by hashing. Also, $j+1$ cannot double-spend a single hash-chain containing j's identity for transactions with other peers, since they always verify the authenticity of received payments with their own identity. Even if colluding peers forge nonexistent relaying and payments, their overall wealth does not increase. Therefore, selfish peers have no incentives to collude with other peers and risk their own privacy and wealth.

Furthermore, neither j nor $j+1$ has to contact the non-interactive PKG during a transaction, unless j or $j+1$ aborts the transaction or j wants to claim payments in batch. With the kept p^0_{j+1}, the PKG can easily find the proper amount j should claim by hashing p^i_{j+1} repeatedly alone. If j indicates the end of the transaction (as a courtesy to $j+1$), the balance of this hash-chain, if any, is refunded to $j+1$ immediately (i.e. instant refund), which can be achieved by the PKG hashing $j+1$'s private-key with j's identity repeatedly alone again. If j does not indicate so, the balance will be refunded to $j+1$ by the PKG automatically when the hash-chain expires (expiry refund), which is indicated in $S^{tn}_{pk_{pkg}}$, so j has to claim received payments before expiry.

End-to-end transactions — As shown in Fig. 2(c), when a peer k wants to obtain b, it broadcasts an authenticated solicitation with sequence number sn to its neighbors for the availability of b and the cost of obtaining b in $SLCT\{id_i, sn, id_k, if(b)\}_{pk_k}$, where i is a potential source of b and $if(b)$ is the meta-information about b. A neighboring peer, e.g. j, can repeat the same solicitation authenticated on its own behalf, if it anticipates its relaying profitable. Within a time window, a peer does not respond to a solicitation that is a subset of its own. In Fig. 2(c), two other peers, j' and j'', follow the same procedure as j does. The solicitation repeats recursively and finally arrives at a peer, e.g. i, that has b available and is willing to offer b to j, j', and j''.

Assume j receives offers from i and j'' securely, and finds it costs less to retrieve b from i directly. Based on its profit strategy, j offers b to k at a price p_j in $RSPS\{\{id_i, sn, id_k, if(b)\}_{pk_k}, id_j, p_j\}_{sk_{j,k}}$, which is profitable for j and supposedly acceptable for k. j has to offer b at a competitive price, since there are other peers competing with j; otherwise, k prefers to deal with others at a better price, and j loses the potential profit from k completely.

Within a time window after its solicitation, k decides whether to obtain b from one of its relaying candidates at a price favorable to itself, or just gives up when none of the received offers is affordable. If the first case happens, k follows the designed per-hop transactions with the chosen relaying peer, so do the upstream relaying peers, as shown in Fig. 2(d). Source peer i should prepare b in a proper format for relaying, e.g. $\{\{m_1\}_{pk_i}, \{m_2\}_{pk_i}, \cdots, \{m_n\}_{pk_i}\}$, where $\{\cdot\}_{pk_i}$ implies these messages are protected in an end-to-end manner by i's signature, so downstream peers can verify the relayed messages and compensate upstream peers independently. If i knows all involved downstream peers (e.g. j and k) with a static route, i can apply an onion-like HMAC chain to each message with the shared-key bootstrapped from their identity, i.e. $\{\{\{m_i\}_{sk_{i,k}}\}_{sk_{i,j}}\}$, which can be verified by j using its shared-key with i. j then passes $\{\{m_i\}_{sk_{i,k}}\}$ to k, which can be verified by k. If the second case happens, k can either increase its broadcast radius (in case k has a hostile neighborhood), move to a location closer to i, or solicit b later when b is cached at nearby peers.

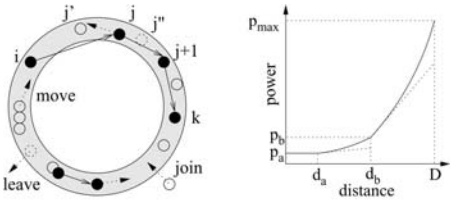
When k obtains b relayed by j from i, it has to pay two types of expense: the cost associated with b (e.g. the cost for i to obtain b, or the value of b assigned by its creator), and the cost to move b from i to k. Here, we decompose end-to-end transactions between the source and destination peer of b into a series of per-hop transactions, so peers only deal with their neighboring peers. Essentially, neighboring peers send upstream peers payments to receive b, and meanwhile receive payments from downstream peers to send b. Since relaying may reduce the overall cost for k to obtain b, k should be willing to share the cost-saving with relaying peers. This *profitability principle* stimulates profitable collaborations among selfish peers in wireless ad hoc networks.

Collaboration strategies — With our security and collaboration measures, peers can have three basic strategies. First, a *voluntary* peer relays for all other peers. Second, a peer is selfish in general, but it becomes *collaborative* only if it is profitably compensated, either by the explicit payment from requesting peers, or by the extracted value of relayed data. Third, a *solely selfish* peer does not relay for others, i.e. it is always non-collaborative.

There are many alternatives to these strategies. For instance, a peer can selectively collaborate with peers that have been collaborative to itself. Also, a peer can choose to follow different strategies throughout its lifetime in the system: initially, it is voluntary when it has plenty of energy; later, it becomes energy-conscious and collaborative only if it is profitably compensated; when its on-board energy is low, it becomes solely selfish and does not relay for others at all. For presentation simplicity, we only consider peers with a chosen strategy throughout their lifetime in a system with different mixes of voluntary, collaborative, and selfish peers in our numerical study.

4 Performance Evaluation

Evaluation approach — We consider a wireless ad hoc network with the topology shown in Fig. 3(a), where N peers are randomly located on a ring of radius



(a) network topology (b) power consumption

Fig. 3. Simulation configurations

#	peer %			demographic
	V	CiC	SS	scenario
I	100	0	0	all voluntary
II	0	0	100	all selfish
III	30	40	30	a general case
IV	0	100	0	all collaborative

Fig. 4. Peer demography

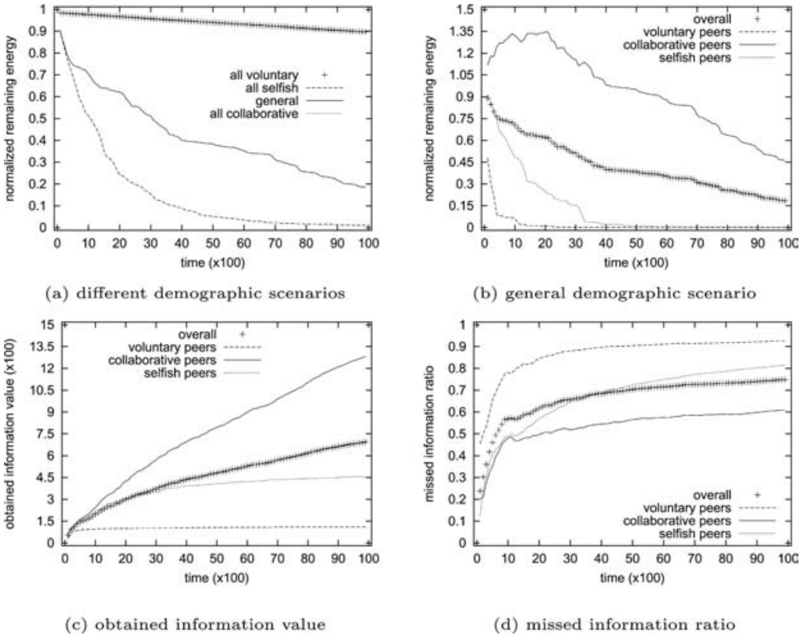
R. With an intentionally-rounded topology, peers have no location disadvantages when compared with peers at any other locations (in contrast, peers close to the border of a finite topology tend to have greater distances to most other peers). This approach allows us to *exclusively* investigate the performance impact of collaboration strategies in wireless ad hoc networks. Arbitrary topologies can be compensated by peer density, preferable pricing, and other means.

When a peer with a certain amount of energy joins the network, it obtains its private-key from the PKG, and deposits a certain amount of monetary credits at the PKG to compensate relaying peers (and the PKG for control traffic). Periodically, peers can trade their accumulated credits for on-board energy (e.g. batteries), and vice versa. The total wealth of a peer is measured by the amount of its remaining energy, available credits, and the value of obtained information. When a peer runs out of energy and credits, it is presumably dead.

Peers can be voluntary (V), collaborative if compensated (CiC), or solely selfish (SS). Their communication cost relies on the distance over which the transferred data cross. As shown in Fig. 3(b), when distance d is less than a threshold d_a, the transmission power consumption remains constant, i.e. $n(d) = 0$ when $0 < d \leq d_a$, and relaying does not offer additional cost-saving. Once $d > d_a$, the distance-related transmission power consumption becomes dominant. When $d_a < d \leq d_b$, relaying is preferable but not critical, e.g. $n(d) = 2$; when $d > d_b$, relaying becomes very attractive by offering significant cost-saving, e.g. $n(d) \geq 4$. The maximum output power ($p_{\max}$) of its transmitter limits the distance that a peer can reach, so $d \leq D$. Here, we set $D < 2R$ intentionally, and peers cannot always communicate with their intended peers directly.

Numerical results — The results presented here are for an IBC-powered wireless ad hoc network of $N = 64$ and $R = 10$. For comparison purpose, all V, CiC, and SS peers have the same amount of initial energy and credits, and are randomly located on the ring. A peer requests data of size 1 to 100 unit length from other peers randomly. We consider four 4 scenarios listed in Tab. 4.

The per-peer performance metrics considered are: the sum of remaining energy and available credits; the volume of obtained information; the volume of missed information due to insufficient energy and credits. From the standpoint of an individual peer, it expects more remaining energy and available credits, more obtained information, and less missed information. For the whole system, the performance metric is the amount of data transferred per unit energy and



(a) different demographic scenarios

(b) general demographic scenario

(c) obtained information value

(d) missed information ratio

Fig. 5. Numerical results for per-peer performance metrics

per unit distance, which reflects the utility of the system employing these measures to facilitate collaboration among peers. We intentionally create an energy-challenged situation by allowing peers to actively request data from other peers (1 request per 100 time unit on average with Poisson distribution), so peers are very likely to miss information due to insufficient energy and/or credits. The *value* of obtained information is weighed by the distance over which the data cross, i.e. if a peer wants to obtain data from a remote peer, it implies that the information is of more value than that from a nearby peer.

In Fig. 5(a), we plot average remaining energy and available credits, normalized to the initial ones, of peers in different demographic scenarios. To avoid warming-up effects, we collect system logs 100 unit time after initialization. Also, results are averaged for 100 runs. When all peers are voluntary (scenario I), a peer always relays packets for others. This is the case when the system is running in the optimal region. On the contrary, when all peers are selfish (scenario II), no peer relays for others, and vice versa; they eventually consume more energy (i.e. a quick drop in remaining energy) to directly communicate with intended peers when feasible, and miss more information when infeasible. When there are certain collaborative peers (about 40% in scenario III), profitable relaying is preferred by these peers, and the overall average remaining energy is considerably higher than that in scenario II. When all peers are collaborative if properly compensated (scenario IV), a significant amount of energy is conserved, and the system is running in a region close to the optimal one in scenario I.

Table 1. System performance metrics

scenario		consumed energy (%)	obtained information	average utility
	overall	81.5	694.508	852.157
III	V	100.0	110.550	110.550
	CiC	53.9	1283.308	2380.905
	S	99.7	455.619	456.990

Fig. 5(b) decomposes the normalized remaining energy for V, CiC, and SS peers in the general demographic scenario (i.e. scenario III). V peers always relay for others, no matter compensated or not; they will attract a lot of selfish peers, and their on-board energy is quickly dissipated. As seen in this figure, V peers almost run out of energy within the first quarter of simulation, due to heavy relaying requests from their neighbors. SS peers take advantage of their neighboring V peers aggressively, so initially their remaining energy reduces relatively slowly when there are many V peers around. When V peers are out of energy, SS peers no longer have free ride. Even worse, since V peers are more likely out of energy when they have SS neighbors, these SS peers eventually pay higher cost to transfer data over greater distances. CiC peers, on the other hand, accumulate credits when they relay for others, and purchase energy when necessary; they conserve energy much better than V and SS peers.

In Fig. 5(c), we show the obtained information value for V, CiC, and SS peers in scenario III. Due to their capability to make profit by relaying packets for others and to conserve remaining energy, CiC peers obtain much more information than V and SS peers. Combining Fig. 5(b) and Fig. 5(c), it is easy to see peer collaboration increases system utility with more obtained information and less consumed energy. This observation is confirmed by the numbers listed in Tab. 1. On average, a unit of energy can transfer about 852.157 unit of data across unit distance for all peers in the system. V peers have the lowest utility of 110.157. Although S peers have a higher utility of 456.990 than V peers by taking advantage of nearby V peers, it is still very surprising to see that S peers indeed have a much lower utility than CiC peers.

Fig. 5(d) gives the ratio of missed information value among all requested information. As we mentioned, we stretch the capability of collaboration schemes in a severely energy-challenged situation, and peers are very likely to miss information. However, as we can see in this figure, CiC peers still outperform V and SS peers. For V peers, their energy is more likely consumed by relaying for others, so they suffer the highest miss ratio. SS peers take advantage of V peers, and have similar performance with CiC peers initially when there are many V peers around. Once most V peers are out of energy, SS peers suffer a much higher information miss ratio as well.

Through these studies, it is concluded that when security and collaboration measures are properly enforced, profitable collaboration is a preferable strategy for all peers in wireless ad hoc networks. Also, with profitable collaboration, system utility increases when peers have maximized their potential profit, which motivates wireless ad hoc networks to adopt these measures.

5 Related Work

Wireless ad hoc networks have attracted intensive attention in recent years [1,2,3, 4]. Their intrinsic vulnerabilities due to the lack of communication and security infrastructures, secured media, trusted peers, and stable states have geared a considerable amount of research efforts toward securing information exchange in these systems [14, 18, 20, 21, 22, 23, 24, 25]. Also, the assumption of voluntary collaboration in wireless ad hoc networks begins to be challenged.

Watchdog and pathrater with overhearing are proposed in [5] to identify peers that agree but fail to forward packets. A majority voting scheme is proposed in [6] to identify misbehaviors by consensus. Packet purse model (PPM) and packet trade model (PTM) [7] use tamper-resistant hardware to circulate and exchange nuglets (a virtual currency). A reputation-based scheme is proposed in [8] to identify and isolate misbehaving peers. CORE also employs watchdog, but has a more sophisticated reputation system to differentiate subjective, indirect, and functional reputation [9]. In Sprite [11], relaying peers keep hashed receipts of forwarded messages, and later claim credits from a central authority when a fast connection is available. Besides fully-distributed wireless ad hoc networks, peer collaboration is also studied in multi-hop cellular systems, where base-stations are available to facilitate and reward collaborative peers. A lottery-like scheme is proposed in [10], where a payee only needs to claim a few winning tickets [26]. A charging and rewarding scheme [12] takes advantage of a trusted base-station that is always involved in communications between any two peers.

In contrast, our hash-chain-based micropayment scheme focuses on *profitable* collaboration among *selfish* peers. It does not use any tamper-resistant hardware, nor does it require an online, interactive authority to be involved in every communication and payment activity. Instead, it explores the profitability principle in packet relaying, and decomposes end-to-end transactions into a manageable series of per-hop transactions. The payment scheme is lightweight, allows intra-payer payment aggregation, and is cheat-resistant against false claim, payment refusal, overspending, and double-spending. Our scheme furthers the idea of PTM [7], without introducing too much network overhead and extra hardware. Our scheme is based on an idea in PayWord [27], but our unique hash-chain construction (i.e. depending on the payer's private-key and the payee's identity) takes full advantage of IBC-powered wireless ad hoc networks, where identity usually is the only means to identify peers, and peer secrecy and wealth are all based on the extracted private-key. An IBC and threshold-based key distribution scheme is briefly outlined in [28] independently, but our work focuses more on peer collaboration rather than key distribution. Also, IBC-based schemes are considered in other contexts such as grid computing [29].

6 Conclusions

Peer collaborations is essential in wireless ad hoc networks due to the lack of infrastructure support; however, voluntary collaboration is found to be too optimistic in practice. In this paper, based on the latest advances in IBC to ensure information confidentiality, integrity, and authenticity, we have designed a hash-chain-based micropayment scheme to stimulate and compensate collaborative peers. The profitability principle and the decomposition approach are generic, and can be applied to other contexts. Our future work will focus on the competitive pricing of selfish peers, especially when relayed data are cacheable at relaying peers for future requests from other peers.

References

1. C. Perkins (ed). *Ad hoc networking.* Addison-Wesley, 2001.
2. Z. Haas, J. Deng, B. Liang, P. Papadimitatos, and S. Sajama. Wireless ad hoc networks. in J. Proakis (ed) *Encyclopedia of Telecommunications*, 2002.
3. R. Ramanathan and J. Redi. A brief overview of ad hoc networks: challenges and directions. *IEEE Comm. Magazine*, 40(5):20–22, 2002.
4. Z. Haas, M. Gerla, D. Johnson, C. Perkins, M. Pursley, M Steenstrup, and C.-K. Toh (eds). Special issue on wireless ad hoc networks. *IEEE Journal on Selected Areas in Comm.*, 17(8), 1999.
5. S. Micali, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proc. 6th ACM Mobile Comp. & Netw. (MobiCom)*, pp. 255–265, 2000.
6. Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. *Proc. 6th ACM MobiCom*, pp. 275–283, 2000.
7. L. Buttyan and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. *Proc. 1st ACM Ad Hoc Netw. & Comp. (MobiHoc)*, pp. 87–96, 2000.
8. S. Buchegger and J. Le Boudec. Performance analysis of the confidant protocol: cooperation of nodes - fairness in distributed ad hoc networks. *Proc. 3rd ACM MobiHoc*, pp. 226–236, 2002.
9. P. Michiardi and R. Movla. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. *Proc. 6th IFIP Conf. on Comm. & Multimedia Security*, pp. 107–121, 2002.
10. M. Jakobsson, J.-P. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. *Proc. 7th IFCA Financial Cryptography (FC'03)*, 2003.
11. S. Zhong, J. Chen, and Y. Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. *Proc. of 22nd IEEE Infocom*, pp. 1987–1997, 2003.
12. N. Salem, L. Buttyan, J.-P. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. *Proc. 4th ACM MobiHoc*, pp. 13–24, 2003.
13. M. Gagnee. Identity-based encryption: a survey. *RSA Labs Cryptobytes*, 6(1):10–19, 2003.
14. L. Buttyan and J.-P. Hubaux (Eds). Report on a working session on security in wireless ad hoc networks. *ACM Mobile Comp. & Comm. Review*, 7(1):74–94, 2003.
15. A. Shamir. Identity-based cryptosystems and signature schemes. *Proc. 4th IACR Conf. on Cryptology (Crypto'84)*, pp. 47–53, 1984.
16. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *Proc. 21st IACR Crypto*, pp. 213–229, 2001.
17. L. Cai, J. Pan, X. Shen, and J.W. Mark. Prompting identity-based key management in wireless ad hoc networks. `http://bbcr.uwaterloo.ca/~cai/tr-ibc.pdf`, 2003.

18. L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.
19. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. *Proc. 3rd IACR AsiaCrypt*, pp. 548–566, 2002.
20. J.-P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. *Proc. 2nd ACM MobiHoc*, pp. 146–155, 2001.
21. G. Montenegro and C. Castelluccia. Statistically unique and cryptographically verifiable (SUCV) identifiers and addresses. *Proc. 9th ISOC Netw. & Dist. Syst. Security (NDSS)*, 2002.
22. Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Proc. 8th ACM MobiCom*, pp. 12–23, 2002.
23. P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. *Proc. SCS Comm. Netw. & Dist. Syst. (CNDS)*, 2002.
24. Y.-C. Hu, D. Johnson, and A. Perrig. SEAD: secure efficient distance vector routing in mobile wireless ad hoc networks. *Proc. 4th IEEE Mobile Comp. Syst. & Appl. (WMCSA)*, pp. 3–13, 2002.
25. K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. *Proc. 10th IEEE Netw. Prot. (ICNP)*, pp. 78–87, 2002.
26. R. Rivest. Electronic lottery tickets as micropayments. *Proc. 1st IFCA FC*, 1997.
27. R. Rivest and A. Shamir. PayWord and MicroMint: two simple micropayment schemes. *Proc. Int'l Workshop on Security Prot.*, pp. 69–87, 1997.
28. A. Khalili, J. Katz, and W. Arbaugh. Toward secure key distribution in truly ad-hoc networks. *Proc. IEEE Security & Assurance in Ad-Hoc Netw. (SAINT)*, pp. 342–346, 2003.
29. T. Stading. Secure communication in a distributed system using identity based encryption. *Proc. 3rd IEEE/ACM Clus. Comp. & Grid (CCGRID)*, pp. 414–420, 2003.

Fireworks: An Adaptive Group Communications Protocol for Mobile Ad Hoc Networks

Lap Kong Law, Srikanth V. Krishnamurthy, and Michalis Faloutsos*

Department of Computer Science & Engineering,
University of California,
Riverside, CA 92521, USA
{lklaw, krish, michalis}@cs.ucr.edu

Abstract. In group communications, we find that current multicast protocols are far from "one size fits all": they are typically geared towards and optimized for particular scenarios. As a result, when deployed in different scenarios, their performance and overhead often degrades significantly. A common problem is that most of these protocols incur high overheads with a high density of group members and in high mobility. Our objective is to design a protocol that *adapts* in response to the dynamics of the network. In particular, our objective is to provide efficient and lightweight multicast data dissemination irrespective of the density of group members and node density. Our work is motivated by two observations. First, broadcasting in some cases is more efficient than multicasting. Second, member and node layout distributions are not necessarily homogeneous. For example, many MANET applications result in a topological clustering of group members that move together. Thus, we develop **Fireworks**, an adaptive approach for group communications in mobile ad hoc networks. Fireworks is a hybrid two-tier multicast/broadcast protocol that adapts to maintain performance given the dynamics of the network topology and group density. In a nutshell, our protocol creates pockets of broadcast distribution in areas with many members, while it develops a multicast backbone to interconnect these dense pockets. Fireworks offers packet delivery statistics comparable to that of a pure multicast scheme but with significantly lower overheads.

1 Introduction

Nodes can be distributed and move in clustered patterns in several MANET applications such as disaster recovery missions and military operations. We believe that this formation of the clustered topology could potentially be exploited to

* This work was supported from grants from NSF CAREER grant no.: 9985195 and prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 853–868, 2005.

reduce the overhead incurred in multicasting. Such clustered topologies are typically characterized by sets of densely packed multicast group members within localized regions. Thus, one could simplify routing by regarding these sets as independent routing entities.

Most of the existing protocols that have been developed thus far [1, 2, 3, 8, 9, 10] do not take the affinity of group members into consideration when constructing their multicast delivery structure. A common problem with these protocols is that the control overhead could be unreasonably large, when the network manifests a dense distribution of group members globally or locally. The control overhead can be high when all the group members are required to participate in the construction and maintenance of the multicast structure. Furthermore, in a mobile environment, a large structure can be frequently under repair[1].

This work is motivated by two observations. First, we observe that a simple broadcast scheme can significantly reduce the control overhead in scenarios wherein the density of group members is high[4]. As a rule of thumb, broadcasting seems more efficient when 40% or more of the nodes in the network are group members. Second, many current protocols cannot adapt to local variations in network properties. Most of these protocols have static, globally pre-defined, parameters that cannot be adjusted dynamically within localized regimes. Our objective then is to design a new protocol that (a) exploits the advantages of broadcasting in high densities and (b) provides localized flexibility in response to changing network conditions.

We propose *Fireworks*, an adaptive multicast/broadcast protocol that exploits group members affinity to simplify routing and invoke broadcast operations in appropriate localized regimes. Fireworks dynamically identifies and organizes the group members into *cohorts* which correspond to areas of high group member affinity. In each of these "dense" neighborhoods, one of the group members is selected to be a *cohort leader*. Cohort leaders have two main functions: (a) they establish a sparse multicast tree among themselves and the source, and (b) they use broadcasting (with adaptive scope) to deliver the packets to other group members in their cohort.

The advantages of this approach are the high adaptability to local properties leading to significantly reduced overheads. This is achieved for the following three reasons: (a) Fireworks reduces the number of group members that participate in the formation and maintenance of the multicast structure and in turn lowers the control overhead, (b) the use of broadcasting in the cohort region maximizes the "wireless broadcast advantage"[2] [13], and (c) the local broadcasts are resistant to changes in the local neighborhood due to mobility.

We perform extensive simulations to evaluate the performance of Fireworks. We study a wide range of scenarios by varying the group sizes, the node mobilities, the number of sources, the traffic load, and the spatial distributions of group members in order to understand the limits of the performance of group commu-

[1] We discuss related work on efforts that address scalable multicasting in section 4.

[2] The fact that every transmission is a broadcast and thus heard by all neighbors.

nications. More specifically, we compare the performance of Fireworks with that of ODMRP[2]. ODMRP has been shown previously to compare favorably with most previously proposed multicast protocols[5]. Fireworks is shown to perform comparably with ODMRP in terms of the packet delivery statistics but with significantly lower overheads. In the presence of multiple group communications sessions or other unicast connections in the network (as the case would be in typical deployment scenarios), this reduction in overheads is especially desirable.

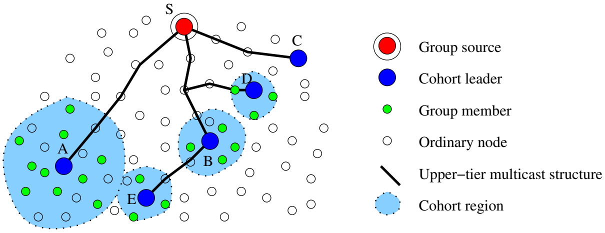


Fig. 1. Fireworks 2-tier multicast hierarchy structure

The rest of the paper is organized as follows: In the next section, we provide a detailed description of our proposed protocol. In section 3, we present our simulation framework and discuss the observed results. Comparisons with ODRMP are also deliberated in this section. We discuss some related works in section 4 and conclude the paper in section 5.

2 Protocol Description

Fireworks, as its name implies, forms a *fireworks-like*[3] group communications structure for data packet delivery. Specifically, it constructs a 2-tier hierarchical structure (see Fig. 1) where the *upper tier* is formed by a multicast source (S in Fig. 1) and cohort leaders (A-E in Fig. 1) that represent groups of multicast members that form a *cohort*, and the *lower tier* consists of the members in a cohort. Since each cohort demonstrates a high density of group members, a cohort leader simply invokes an adaptive localized broadcast within its cohort to disseminate multicast packets received from the source. This would reduce the consumed overhead while ensuring efficient data delivery as observed in [4].

In the following subsections, we describe the details of Fireworks. We deliberate on the construction of the two tier hierarchy and the actual use of the construction for data delivery.

[3] The transmission of data packets from the source to cohort leaders is analogous to emission of firework shells to some predefined spots in the sky; the broadcast of data packets by each leader in the cohort is analogous to the explosion of fireworks at the predefined spots.

2.1 Definitions of Protocol States and Data Structures

Prior to our detailed discussion of Fireworks, we define a few data structures and protocol states that are employed. These definitions are used throughout the latter sub-sections that detail protocol operations.

1. **Role**(*role*). Each group member in Fireworks has a *role*: it could either be in a transient mode wherein it is *JOINING* the session, could be a cohort *LEADER* or could simply be the *CHILD* of a cohort leader.
2. **MGroup**(*mg*). This state variable, maintained by each group member, indicates the current multicast group of the group member.
3. **Leader**(*ldr*). This variable maintains the address of the cohort leader with which the group member is affiliated (if the group member is a child). If the group member is a cohort leader itself, this value is set to NULL.
4. **Distance**(*d*). The distance to the cohort leader is maintained by this state variable. If the group member is a cohort leader itself, this value could simply set to a very high value (i.e. infinity).
5. **Cohesiveness**(*c*). This is a state variable that maintains the affinity of group members within a node's k-hop[4] radius; it is computed as follows:

   The cohesiveness of a node, say i, is defined as:

   $$c_i = \sum_{\forall n \in N_i^k} (k - distance_{i,n} + 1) \tag{1}$$

   where N_i^k is the set of group members that are within a k-hop radius from node i; the $distance_{i,n}$ is the hop distance from node i to node n. The higher the number and the closer the group members in its proximity, the greater the cohesiveness of a node has.
6. **Join Group Table** (*JGTable*). This table, maintained at each node, maintains information with regard to the *JOINING* group members and the existing cohort leaders that are nearby. Each entry in the table contains the *address*, *mcast-address*, *role*, *distance* and *cohesiveness* as it pertains to the nearby group member or cohort leader. The information maintained in this table is obtained by means of the ADVERTISE and the LEADER messages (to be discussed in section 2.2).
7. **Cohort Member Table** (*CMTable*). This table is maintained *only* by cohort leaders. It maintains information with regard to all the group members of the cohort (called children or cohort members) that are associated with the cohort leader. Each entry in the table contains the *address*, *mcast-address*, and the *distance* of each child. The information is obtained via the reception of CHILD messages that are sent out by each cohort member.

[4] k is a system parameter. We consider the case when $k = 2$ since it gives the optimal trade-offs between performance and overhead.

2.2 Construction of the Fireworks Multicast Structure

The construction of our *fireworks-like* structure consists of three steps: (1) The determination of roles by group members, (2) the creation of the upper tier multicast structure, and (3) the employment of adaptive broadcast in the lower tier multicast structure (i.e. within a cohort). These steps are described below:

Role Determination of Group Members. The determination of the role of a group member is composed of two phases:

1. **Discovery Phase.** In this phase, the joining node discovers the other joining group members and cohort leaders in its vicinity. When a node decides to join a multicast group, it enters this phase and advertises its presence to its k-hop neighborhood by broadcasting an ADVERTISE message. The ADVERTISE message has a scope of k hops and contains the *address*, *mcast-address*, *hopcount* and *cohesiveness* of the node. Upon the reception of an unique ADVERTISE message, nodes update their *JGTable* as per the contents in the message. After this phase, each joining node would have obtained the k-hop local topology information (in the absence of packet losses). This information is used (if needed) in the decision phase (to be discussed) to determine the cohort leaders. Packet losses can result in a reduction in the accuracy of the topology information. However, our studies show that due to the inherent redundancy provided by broadcasting, such losses are rare and have negligible effects on the performance of Fireworks. This phase may be triggered again when the connection to the cohort leader is lost.
2. **Decision Phase.** In this phase, the joining node determines if it should choose to be the cohort leader for its k-hop neighborhood. If after the discovery phase, if a joining node cannot still find any cohort leader in its vicinity, it will enter this phase. [5] If its cohesiveness value is the highest as compared to its k-hop neighbors[6], it will elect itself as a cohort leader and serve a cohort. It then changes its *role* to *LEADER* and broadcasts a LEADER message containing its *address*, *mcast-address*, *cohesiveness* and *hopcount*. The TTL value of this message is set to k so as to notify the node's k-hop neighbors of the presence of a new cohort leader[7]. Nodes that are within the broadcast scope of the LEADER message update their *JGTable* to reflect the content of the message.

[5] Note that the first decision phase (during initialization) is started after at least two ADVERTISE messages have been sent. This is due to the fact that the first ADVERTISE message initially has a cohesiveness value of zero since, in the beginning, nodes are unaware of their neighborhoods.

[6] This is indicated by the entries built up in its *JGTable*.

[7] As discussed later, the distribution scope of the subsequence LEADER messages could be dynamically adjusted.

During these phases, a joining node may receive several LEADER messages. If this is the case, the joining node will pick the best cohort leader to join[8] by unicasting a CHILD message containing its *address*, *mcast-address* and *hopcount* to the selected cohort leader to notify the cohort leader of its intention to join the cohort. The cohort leader would then update its *CMTable* accordingly.

Note that if a joining node is unable to find any cohort leader in its vicinity *and* based on the above criteria is unable to elect itself as a cohort leader, it will invoke additional instances of the discovery and the decision phases. Consequently, after the completion of the above phases, a joining node *must* either become a cohort leader or a child of a cohort leader. From then on, each cohort formed becomes a single routing entity as represented by its cohort leader. Only the relatively small number of cohort leaders will then participate in the construction and maintenance of the multicast structure.

Creation of Upper Tier Multicast Structure. To enable the construction of the upper tier of the Fireworks multicast structure, the multicast source periodically broadcasts a SOURCE-QUERY message containing its *address* and *mcast-group* to the network. Intermediate nodes forward unique SOURCE-QUERY messages further and set up pointers backward towards the source. When a cohort leader receives the SOURCE-QUERY message, it unicasts a SOURCE-REPLY message back to the source via the route established by the aforementioned backward pointers. The nodes along the unicast path towards the source become the forwarding nodes for the group and are identified by the (*source, mcast-group*) attribute pair. From then on, data packets are multicast from the source to the cohort leaders via a tree constructed by coalescing the constructed reverse unicast paths. Note that this is not a source tree. Forwarding nodes, upon the receipt of SOURCE-REPLY from more than one cohort leader, conclude that they are the root of a multicast sub-tree and forward packets to their multiple children on the tree.

Adaptive Broadcast Within Cohort. Once the cohort leader receives a data packet from the source, it performs a broadcast within its cohort to deliver the data packet to the associated group members. Note that the broadcast operation performed is adaptive in the sense that the maximum broadcast scope is not simply set to k hops but instead depends on the furthest child of the cohort leader. In other words, the broadcast scope could be reduced as per the *distance* information of each furthest child which is contained in the *CMTable*. This adaptability could reduce unnecessary transmissions of data packets that could result as a result of setting the broadcast scope too large. An example is illustrated in Fig. 1 where cohort leaders may have different broadcast scopes. The cohort leaders (B, D and E) maintain cohorts of radius 1-hop since there are no children that are beyond this distance. In the extreme case when a group

[8] The best cohort leader is the one that has the shortest distance and highest cohesiveness (in that order); further ties are broken by selecting the one with the highest nodeID.

member is isolated (Node C in Fig. 1), the isolated group member will become a cohort leader at the conclusion of the aforementioned phases. Such a singular leader has no children and thus, will not perform any local broadcast.

2.3 Joining a Multicast Group

A node is considered to have joined a multicast group if its role is either that of the cohort leader or if it is deemed a child of a cohort leader. The process of joining a multicast group is described below.

When a node decides to join a multicast group, it simply changes its *role* to *JOINING* and enters the discovery and decision phase as described in section 2.2. If the joining node has cohort leaders in its k-hop vicinity, it would possibly receive LEADER messages before entering the decision phase. If this is the case, the joining node will simply pick the best cohort leader to join (become a child of a cohort leader) as described in section 2.2. If the joining node has no cohort leader present in its vicinity and its cohesiveness is the highest as compared to its k-hop neighbors, it will become a cohort leader and serve a cohort.

2.4 Leaving a Multicast Group

Group members could leave a multicast group at anytime. A group member that has the *role* of *CHILD* simply stops unicasting the CHILD message to its cohort leader. Fireworks is based on maintaining soft-state and after a predefined timeout, entries are purged from the tables listed earlier.

When a cohort leader decides to leave the multicast group, it simply stops transmitting the LEADER message. Cohort members, upon discovering the absence of a leader, will first try to quickly rejoin another cohort by looking for other leaders in their *JGTable*. If no cohort leader is present in a member's vicinity, the cohort member will switch its *role* to *JOINING* and invoke the discovery and decision phases to find another cohort or to become a cohort leader as described in section 2.2.

2.5 Maintaining the Multicast Structure

Due to the node mobility, the upper tier multicast structure and the formation of cohorts will have to be continually updated. We describe below the maintenance functionalities of different entities with Fireworks.

Source Functions. The source periodically refreshes the upper tier multicast structure (the tree to the cohort leaders) by triggering the exchange of SOURCE-QUERY and SOURCE-REPLY messages as described in section 2.2. By means of this, the multicast tree structure might be refined. Stale routes may be purged and new ones created due to changes that occur as a result of mobility.

Cohort Leader Functions. Each cohort leader periodically broadcasts a LEADER message to its cohort. The purpose of this periodic announcement is to indicate its continued existence to the associated cohort members. In addition, this rebroadcast acts as an invitation to the leader's nearby *new* group members that are not currently associated with the cohort. Each cohort member (*role* =*CHILD*) sends updates that contain the distance of the member to its cohort leader regularly (to be discussed in detail). Using this, a cohort leader, is able to dynamically adjust the scope of the local broadcast as mentioned earlier. The broadcast scope of the LEADER message is set to 2 hops if the number of cohort members (as recorded in *CMTable*) and the estimated number of new cohort members (specified in the *JGTable*) together is greater than a predefined threshold[9]. If these conditions do not hold, the LEADER message broadcast scope is set to 1 hop. The reason of reducing the LEADER message broadcast scope is that when the number of cohort members becomes small, the advantages of performing local broadcasts are lost. This reduction of the broadcast scope of the LEADER message to a single hop is akin to simply resorting to unicast transmissions (by using the broadcast channel), from the source to the associated members of the cohort via the leader. Note that in this case, the members are simply a hop away from the cohort leader.

Cohort Member Functions. Each cohort member periodically indicates its existence and updates its distance to its cohort leader so that the cohort leader could dynamically adjust its broadcast scope as discussed previously. This is done by unicasting a CHILD message to the cohort leader. The cohort leader will update its *CMTable* as per the contents of this message. Since the probability of a given cohort member implicitly leaving the associated cohort depends on the member's distance to the cohort leader (i.e., the closer the cohort member to its leader, the less possible it is that it moves out of scope), the frequency of these unicast updates from a member depends on this distance of the member from the leader. Our simulation results show that reducing the update frequency of the 1-hop cohort members has negligible effects on the performance of Fireworks in terms of the packet delivery ratio but significantly reduces the incurred control overhead[10].

Sometimes, a cohort member may overhear LEADER messages of leaders from other cohorts. When this happens, the cohort member will see if the cohort leader that transmits the LEADER message is closer than its current cohort leader. If it is, the cohort member will switch to the new cohort by updating its state variables (*ldr* and *d*) and unicasting a CHILD message to the new cohort leader.

The connection between a cohort member to the cohort leader is deemed lost if the cohort member misses 3 consecutive LEADER messages from the cohort leader (via a time-out that accounts for this). In this case, the disconnected cohort member will, at first, try to rejoin a different cohort by looking for other leaders in its *JGTable*. If other cohort leaders are available, the disconnected cohort member will join the best leader as described in section 2.2. If no leaders

[9] This threshold is set to 5 throughout our evaluations. This has been seen to be an appropriate threshold as per our previous studies [4].

[10] In our evaluations, all 2-hop cohort members update at 3 seconds intervals and all 1-hop cohort members update at 9 seconds interval. Small changes to these values did not cause the performance to change by much.

are found in the table, the disconnected cohort member will try to rejoin the group by invoking the discovery and decision phases as described in section 2.2.

Relinquishing Cohort Leader Functionalities. A cohort leader will give up its *LEADER* role when it determines that it is no longer necessary to maintain itself as a leader. In Fireworks, a cohort leader that has no children is required to regularly check for the presence of other cohort leaders in its vicinity. Upon finding a leader, it will give up its own *LEADER* role and switch to a *CHILD* role by joining the discovered leader.

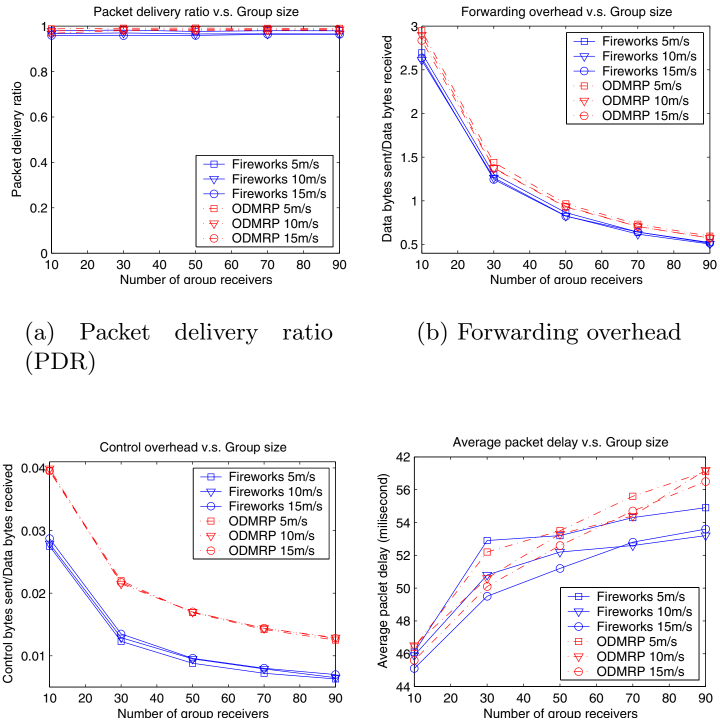
A second scenario that may lead to the relinquishment of cohort leader is when two or more cohort leaders come within the range (within k hops) of each other due to mobility. Even though Fireworks does not strictly enforce the existence of only a single leader within a k hop radius (since this may complicate the operation of Fireworks), cohort leaders may give up their roles if this were to happen. This is because, members tend to migrate to the "best" cohort leader among the cohort leaders that drift together. This may cause some of the cohort leaders under discussion to lose all their cohort members. Such members would then relinquish their *LEADER* roles as discussed earlier.

3 Performance Evaluation

In order to evaluate the performance of Fireworks, we implement and simulate the protocol in NS-2[6] and compare the obtained performance with that of the well-known multicast protocol, the On-Demand Multicast Routing Protocol (ODMRP). We pick ODMRP as a baseline protocol since it has shown to be one of the elite protocols in its class[5].

We divide our evaluations into two parts. In the first part, we evaluate the performance of Fireworks under *randomly constructed network scenarios.* In these scenarios, all nodes are uniformly and randomly distributed throughout the simulation area at the beginning of the simulation. The movements of nodes are guided by the Random Waypoint model. In the second part, our objective is to demonstrate the adaptability of the Fireworks under *clustered network scenarios.* The scenarios in question are similar to the random network scenarios but we intentionally include formations to reflect clustered group members (cohorts) in the networks. The motion of these clustered groups members are defined by the Reference Point Group Mobility (RPGM) model[7]. In this model, logical groups are defined and their movements are correlated with the motion of their so called respective *reference points.* In our evaluation, we pick one node from each logical group to be the *reference node* and its position and speed is used to guide the motion of the members in its logical group.

In the simulations, nodes have a transmission range of 250 meters and a maximum transmission rate of 2Mb/s. The total simulation time is 100 seconds and we repeat the simulations for 40 times and obtain the average results. The first source (randomly chosen among the source nodes) begins the transmission of data at time 20s and if additional sources are present, they start transmitting



(a) Packet delivery ratio (PDR)

(b) Forwarding overhead

(c) Control overhead

(d) Average packet delay

Fig. 2. Comparing the performance of Fireworks and ODMRP under random network scenarios with varying group sizes and mobilities

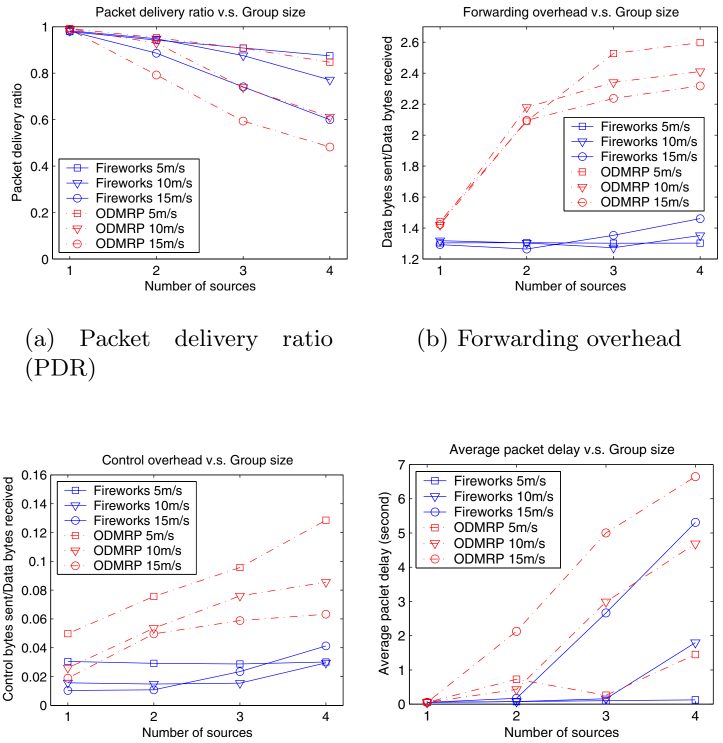
data one after another (again randomly chosen) with the starting instances separated by 0.5s. Group members randomly join the group between [0, number of group members × 0.01) seconds. The data packet size is set to 512 bytes.

3.1 Simulating Random Network Scenarios

In these experiments, the parameters that we vary in order to evaluate the performance of Fireworks under different settings are: *group sizes*, *node mobility*, *number of sources* and *traffic load*. The performance metrics that we are interested in are: *packet delivery ratio*, *data forwarding overhead*, *control overhead* and *average packet delay*.

The common simulation settings that are used in these experiments are the simulation area (1250m×1250m), the number of nodes (100) and the number of multicast groups (1).

Simulation 1: Group Size and Node Mobility. First, we examine the effects of the group sizes and node mobility on the performance of Fireworks and



(a) Packet delivery ratio (PDR)

(b) Forwarding overhead

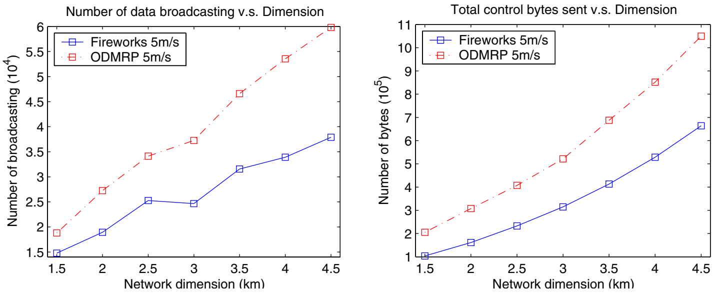
(c) Control overhead

(d) Average packet delay

Fig. 3. Comparing the performance of Fireworks and ODMRP under random network scenarios with varying number of sources and traffic rates

compare the performance with that of ODMRP. The common fixed parameters are the traffic rate (5 pkts/s) and the number of sources (1).

The results of these simulations are shown in Fig. 2. The packet delivery ratios with both Fireworks and ODMRP approach a 100% for all group sizes and node mobilities. In terms of the data forwarding overhead, Fireworks incurs around 10% less overhead as compared to ODMRP. This reduction in data forwarding overhead is due to the simpler multicast structure constructed by Fireworks as compared with ODMRP. Although Fireworks does perform broadcasts within each cohort, the incurred data forwarding overhead is still lower; this in turn implies that performing broadcasting in local cohorts is very effective. In terms of control overhead, Fireworks is the clear winner. Fireworks incurs 30% less overhead when the group size is 10% (10 % of the nodes in the network are group members) and up to 50% less overhead when the group size is increased to 90%. The reduction in control overhead is especially significant when the data packet size is small (comparable to the control packet size), since the total incurred overhead would then be dominated by the control overhead (as opposed to data overhead due to redundant data transmissions). In terms of average packet delay,



(a) Total number of data broadcasting

(b) Total number of control bytes sent

Fig. 4. Comparing the overheads of Fireworks and ODMRP

Fireworks also tends to have lower delay than ODMRP. This could potentially be due to the lower queuing delay thanks to the reduction in the load due to the overhead, with Fireworks.

Simulation 2: Number of Sources and Traffic Load. In this experiment, our objective is to study the effects of the number of sources and traffic load on the performance of Fireworks and compare the performance with that of ODMRP. The common fixed parameters are the number of group members (30) and the node mobility (5m/s).

The simulation results are shown in Fig. 3. With an increase in the number of sources from 1 to 4 and with traffic loads varying from 2pkts/s to 6pkts/s, the packet delivery ratios of both ODMRP and Fireworks decrease. However, as observed, Fireworks is able to maintain a better delivery ratio under higher traffic load and with a higher number of sources. This is because Fireworks generates, in general, lower data forwarding and control message overhead than ODMRP. The contention and therefore, collisions are thus less severe in Fireworks enabled networks than in ODMRP enabled networks. This is elucidated in Figures 3(b) and 3(c). The data forwarding overhead and control overhead are much lower with Fireworks than with ODMRP in all scenarios considered. ODMRP creates a group-based mesh, and the excessive redundancy created by the mesh is not seen in Fireworks as the created forwarding nodes are attributed by a specific (source, mcast-group) pair. Furthermore, the cohorts formed in Fireworks are shared between all the sources of the same group and thus the control overhead incurred by the cohorts will not be affected by the number of sources. Note that due to the lower congestion level in the network, the average packet delay incurred by using Fireworks is much smaller than that of ODMRP in all scenarios.

3.2 Simulating Clustered Network Scenarios

In these experiments, we want to further emphasize the benefits Fireworks can offer due to its having considered group member affinity in constructing the multicast structure. In the scenarios considered, clustered group members are

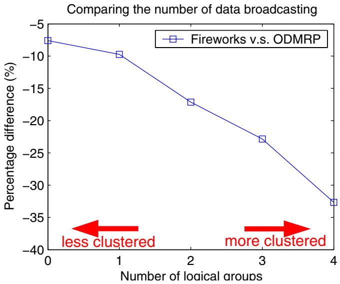


Fig. 5. Comparing the adaptability of FIREOWKRS to ODMRP

introduced as discussed earlier[11]. The main parameters of interest are (i) the size of the network (in terms of network dimension or number of nodes) and the correlation between group members' motion patterns. We enumerate the performance of Fireworks in terms of the reduction in overhead as compared with ODMRP. The performance metrics that we consider are *the number of data forwarding* instances and *the number of control bytes* incurred.

Some common simulation parameters that are relevant to these experiments are the node mobility (5m/s), the number of groups (1) and the number of sources (1).

Simulation 3: Varying the Network Size. In this experiment, we would like to see the effects of varying the network size on the performance of Fireworks and ODMRP. The common fixed parameters are the traffic rate (5 pkts/s) and the group size (40). We introduce 2 logical groups, each with 20 group members within a circular area of 400m radius. The number of nodes increases when the physical network size increases such that the density of nodes is maintained. The number of nodes under various physical network sizes are: 180 in 1.5km^2, 320 in 2.0km^2, 500 in 2.5km^2, 720 in 3.0km^2, 980 in 3.5km^2, 1280 in 4.0km^2 and 1620 in 4.5km^2.

The results are shown in Fig. 4. As we see, increasing the physical network size (number of nodes) increases the number of data broadcasting performed and the amount of control bytes sent in both Fireworks and ODMRP. However, Fireworks has much smaller number of data broadcasting and control bytes sent as compared to ODMRP. These results indicate that Fireworks is able to adapt the environment better by identifying the logical groups and appropriately constructing fewer routes that are targeted towards the groups. As the network size increases, the average path length from the source to each multicast destination increases and treating each destination independently to construct a mesh (as with ODMRP) can lead to increased overhead.

[11] As mentioned, this could represent a group of firefighters or rescue workers operating in the proximity of each other or a team of soldiers in a tactical mission.

Simulation 4: Examining Clustered Motion. In this evaluation, we examine the effects of clustered motion (as discussed) on Fireworks and ODMRP. The fixed parameters are the simulation area (2000m×2000m), the number of nodes (300), the traffic rate (2 pkts/s) and the group size (40). We increase the number of logical clustered groups from 0 to 4. Each logical group consists of 10 group members and these group members move as per the RPGM model. For those group members that are not in any logical group, the motion is as per the random waypoint model.

The results of the simulations are shown in Fig. 5. The results represent the percentage differences in the number of data broadcasting instances between Fireworks and ODMRP. As more logical groups are defined, the network becomes more clustered which means that group members move together (motion is correlated). We see that Fireworks is able to adapt to clustered motion far better than ODMRP due to its inherent features as discussed earlier.

4 Related Works

Numerous multicast protocols have been developed for use in MANETs. MAODV[1] is a multicast extension of its unicast counterpart. ODMRP[2] is a mesh-based multicast protocol which creates a mesh structure for reliable data delivery. CAMP[8] constructs a group-shared mesh which makes use of a core node to reduce the control traffic needed for receivers to join group. AMRIS[9] makes use of ID number to guide the construction of a tree-based shared multicast structure which supports multiple senders and receivers. AMRoute[10] is a hybrid multicast protocol which constructs a virtual multicast tree on top of the virtual mesh links established between group members. All of these protocols create a flat routing topology and are unaware of the topological characteristics of the structure. Unlike Fireworks, none of the previous schemes adopt broadcast features to adapt to local conditions.

Recently, a hierarchical multicast protocol called HDDM has been proposed in [11]. It is targeted to provide scalable multicasting in MANETs. The idea of the protocol is to extend the scalability of the DDM[12] multicast protocol which was used to support multicasting in small groups. The protocol divides the whole network into different sub-groups by selecting suitable sub-roots that are responsible for delivering data packets using DDM protocol to their respective sub-group members. While HDDM requires the source to have a complete list of group members and requires an underlying unicast protocol to provide routing information, Fireworks does not. The unicast routing information is used by the HDDM source to determine its sub-roots. Each sub-group is basically a multicast tree that consists of sub-group members rooted at a selected sub-root. Although Fireworks constructs a hierarchical structure, the criteria for the creation of the tiers and the purpose of the sub-groups (cohorts in Fireworks) are substantially different in the two protocols. Fireworks constructs cohorts based on group member affinity which aims at maximizing the wireless broadcast advantage. HDDM aims at providing a suitable sized sub-group for efficient DDM protocol deployment.

5 Conclusions

In this paper, we propose a new hybrid multicast/broadcast scheme for MANETs. The construction is primarily geared towards reducing overheads incurred with group communications in MANETs. Fireworks exploits the property that the use of a broadcast scheme in an area of densely distributed group members could significantly reduce protocol overhead. It takes the group members affinity into account in constructing the data delivery structure and dynamically partitions a multicast group into several smaller cohorts in such a way that the formed cohorts manifest a high level of group affinity. A simple broadcast scheme is then used to provide a low-overhead data delivery service within these cohorts. From our simulation results, the *fireworks-like* data delivery structure constructed is shown to be lightweight in terms of the control and data forwarding overheads of the protocol. Since Fireworks employs broadcasting within a cohort, the inherent redundancy provides reliability and a packet delivery performance that is comparable with that of a pure multicast protocol (ODMRP) is achieved. Even though Fireworks is specially designed for clustered networks, our results also demonstrate its superior performance as compared with ODMRP under random network deployment scenarios as well.

Note: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

References

1. E.M. Royer and C.E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in Proceedings of the 5th annual ACM/IEEE MOBICOM, 1999, pp. 207-218.
2. S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-Demand Multicast Routing Protocol," in Proceedings of IEEE WCNC, 1999, pp. 1298-1304.
3. C.-C. Chiang, M. Gerla, and L. Zhang, "Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks," in Cluster Computing, vol.1, no.2, 1998, pp. 187-196.
4. L.K. Law, S.V. Krishnamurthy, and M. Faloutsos, "On Evaluating the Trade-offs between Broadcasting and Multicasting in Ad Hoc Networks," in Proceedings of the IEEE MILCOM, 2004.
5. S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols," in Proceedings of IEEE INFOCOM, 2000, pp. 565-574.
6. S. McCanne and S. Floyd, "Ns-2 simulator." http://www.isi.edu/nsnam/ns/
7. X. Hong, M. Gerla, G. Pei and C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," in Proceedings of ACM/IEEE MSWiM, 1999, pp. 53-60.
8. E.L. Madruga and J.J. Garcia-Luna-Aceves, "Scalable multicasting: The Core-Assisted Mesh Protocol," ACM/Baltzer Mobile Networks and Applications, Special Issue on Management of Mobility, vol.6, no.2, pp. 151-165, Apr. 2001.
9. C. Wu and Y. Tay, "AMRIS: A Mulitcast Protocol for Ad Hoc Wireless Networks," in Proceedings of IEEE MILCOM'99, Atlantic City, NJ, Nov. 1999.

10. J. Xie, R.R. Talpade, A. Mcauley, and M. Liu, "AMRoute: Ad Hoc Multicast Routing Protocol," Mobile Networks and Applications., vol.7, no.6, pp. 429-439, 2002.
11. C. Gui and P. Mohapatra, "Scalable Multicasting in Mobile Ad Hoc Networks," in Proceedings of IEEE INFOCOM, 2004, Hong Kong.
12. L. Ji and M.S. Corson, "Differential Destination Multicast - A MANET Multicast Routing Protocol for Small Groups," in Proceedings of IEEE INFOCOM, 2001, Anchorage, Alaska.
13. J. Wieselthier, G. Nguyen, and A. Ephremides, "On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks," in Proceedings of IEEE INFOCOM, 2000.

An Optimized TCP for Internet Access of Vehicular Ad Hoc Networks

Marc Bechler, Sven Jaap, and Lars Wolf

Technical University of Braunschweig,
Institute of Operating Systems and Computer Networks,
{bechler, jaap, wolf}@ibr.cs.tu-bs.de

Abstract. Communication efficiency at the transport layer is of specific importance for ad hoc networks. Especially in vehicular ad hoc networks, vehicles will have a temporary and rather short-lived connectivity to the Internet, which has to be utilized efficiently. In this paper, we propose a TCP-based transport protocol called MCTP that is optimized for the Internet access in vehicular environments. Therefore, MCTP is combined with split performance enhancing proxy architectures, where a proxy separates the end-to-end TCP connection. This enables the deployment of optimized transport protocols while maintaining interoperability with TCP used in the Internet. For the evaluation, we emulated the communication characteristics of a "typical" vehicular scenario. This clearly shows the advantages of MCTP over traditional approaches; the overall data throughput is significantly higher when MCTP is used for communication between vehicle and proxy. The evaluation also emphasizes the usefulness of performance enhancing proxies in vehicular environments.

1 Introduction

Communication in vehicular environments will become very important and crucial for the future development in the automotive domain: it is considered as a key technology to increase traffic safety since vehicles will be able to distribute local information to other vehicles on the road. For example, emergency situations like an accident or a congestion behind a bend can be transmitted to succeeding vehicles. This way, the vehicles are able to slow down their speed in time. A key technology for inter-vehicle communication (IVC) is multi-hop ad hoc networking. Thereby, vehicles establish vehicular ad hoc networks (VANETs), which enable the local exchange of information without the need for infrastructure components like base stations. Examples for IVC systems are the FleetNet communication system [1] or CarNet [2].

With the introduction of VANETs, passengers also expect infotainment services as well as the access to Internet services using the IVC system. The transition between vehicles and the Internet is achieved by gateways installed on the road-side. The gateways thus provide a temporarily restricted access to the Internet for the passing vehicles traveling in a (specified) area around the gateways. Application scenarios are manifold, as illustrated by the following examples:

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 869–880, 2005.

- businessmen likely want to send and download emails, and they may synchronize their personal information applications with their office systems,
- the navigation unit of a truck may want to communicate with the company's fleet management system in order to exchange time sensitive information.

In order to access Internet services, VANETs must be integrated into the Internet. This integration is typically achieved by performance enhancing proxies. For example, fig. 1 depicts the proxy architecture used for the Internet integration of the FleetNet IVC system [3]. Thereby, the VANET has connectivity to the Internet through gateways, which are itself connected to a gateway network. A proxy located at a fixed position in the Internet hides the characteristics of the VANETs and, thus, brings together the VANET and the Internet. The proxy also separates the end-to-end TCP connection into two segments: communication between proxy and Internet hosts using standard TCP, and communication between vehicles and proxy. This way, highly optimized transport protocols can be used for communication between proxy and vehicles in order to improve communication efficiency.

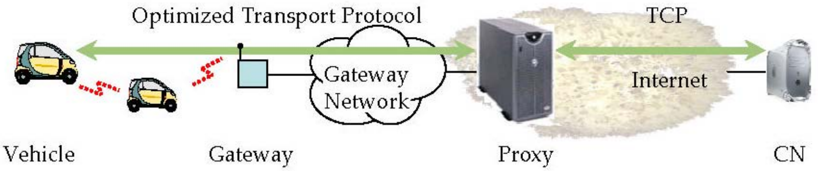


Fig. 1. Vehicular communication scenario

In this paper, we propose a TCP-based transport protocol called MCTP (Mobile Control Transport Protocol), which is optimized for proxy-based communication architectures used in vehicular environments. We describe the basic protocol mechanisms used in MCTP and compare its performance with traditional approaches in a test environment that emulates the characteristics of a typical communication scenario on a highway.

In the following, we first describe related work on improving TCP performance in section 2. Section 3 introduces our transport protocol MCTP, which is evaluated in section 4. Finally, section 5 concludes this paper.

2 Related Work

TCP was developed for networks with a fixed topology. This way, it works well in wired networks and provides an acceptable performance in terms of data throughput. However, the characteristics of mobile networks like VANETs differ fundamentally from wired networks: On the one hand, vehicles are highly mobile and therefore the topology of the VANET is subject of permanent reconfigurations and partitionings. On the other hand, communication is based on wireless radio technology, which shows high variations in the transmission quality. Internet access also will not be available continuously resulting in potentially long periods of disconnections. Several studies investigated the impact of these aspects on the performance of TCP. The investigations showed that TCP provides poor throughput in multi-hop ad hoc networks although a higher throughput might be possible in theory [4]. The performance degradation mainly results from the conservative flow and congestion control mechanisms deployed in TCP. For example, TCP interprets transmission errors as a congestion situation and thus reduces the throughput. The algorithms used are slow start and congestion avoidance [5]. Over the years, TCP was enhanced by several new protocol features. TCP Reno introduced fast retransmit/fast recovery, which was further improved in TCP New Reno according to RFC 2582. Furthermore, TCP was enhanced by selective acknowledgements (RFC 2018). These extensions are already integrated in TCP implementations of common operating systems like Linux. However, such extensions do not solve the basic problems of TCP in mobile environments. This way, TCP still provides a poor performance in the VANET scenario, i.e. for communication between a vehicle and the proxy [3]. In order to improve end-to-end communication efficiency at the transport layer, related work can be classified into three categories (RFC 2757): (i) pure congestion control modifications, (ii) utilization of information from intermediate systems, and (iii) completely new transport protocols not based on TCP. We do not consider snoop-based approaches since they are not expected to provide significant improvements in networks with a high frequency of handoffs.

An obvious way to increase performance is to modify the congestion control in TCP. A noticeable amount of work tries to predict different situations based on local information. With the help of this information, the congestion control algorithms of TCP are modified to react accordingly depending on the predicted situation. Several approaches like TCP Westwood [6] try to estimate the available bandwidth in an intelligent way, which is used to optimize the TCP flow control. Other approaches like TCP DOOR [7] modify the congestion control based on the arrival of out-of-order packets, or they even examine inter-packet arrival times for using a rate-based congestion control mechanism (e.g., Wireless TCP [8]). Approaches like ADTCP [9] additionally measure short term throughput, packet loss ratio, and packet out-of-order delivery ratio, and they use a modified TCP state machine to react efficiently in these situations. Another common solution is to completely modify the algorithms used for slow start, congestion avoidance, and various timeout calculations like, e.g., TCP Vegas [10]. Approaches like ATP [11] completely replace the congestion control of TCP by different algorithms. In Freeze-TCP [12], the receiver notifies the sender in case of an impending congestion. The sender then “freezes” TCP to prevent further transmissions.

A general drawback of this category is that predictions about potential congestion situations are based on local information, which may not reflect the current state of the VANET. This misprediction potentially reduces TCP performance. Moreover, the congestion control algorithms do not provide mechanisms to handle both short-term and longer-term periods of disconnections.

The second possibility is to utilize information from intermediate systems, if the network is able to detect different situations. A common mechanism is Explicit Congestion Notification (ECN, RFC 3168), where intermediate nodes are able to detect pending congestions and signals them to the communicating end systems. This way, an ECN-enabled TCP may use this information to optimize communication efficiency. The utilization of information from intermediate systems is a promising approach to improve TCP in VANETs. The network information provides a better accuracy of the estimations compared to the predictions of pure congestion control modifications. This concept implicitly includes the consideration of notifications, which enables TCP to react quickly to various situations in the network. However, TCP extensions like ECN basically do not solve the general problems of TCP in VANETs since these approaches are still based on exponential backoff timers to calculate the retransmission timeouts. This mechanism is not suitable to handle long-term disconnections from the Internet appropriately since they may cause either a reset of the TCP connection or a long recovery phase after a reconnection to the Internet.

The third category comprises transport protocols not based on TCP. A typical example is the Stream Control Transmission Protocol (SCTP, RFC 2960). In contrast to TCP, the connection-oriented SCTP supports multi-streaming and multi-homing capabilities. This category is not discussed further on since such protocols do not provide a socket-like API, which requires new network programming paradigms that aggravate the deployment of existing applications in vehicular environments.

3 MCTP

An optimized transport protocol for vehicular environments must be able to distinguish between error-prone links and network congestions in order to handle packet losses appropriately. Moreover, it must be able to utilize information from both intermediate systems and from underlying protocols. This is necessary for an efficient treatment of both short-term network partitions and longer-term periods of disconnections from the Internet. However, none of the existing related work fulfils these requirements sufficiently. This way, we developed the transport protocol MCTP (Mobile Control Transport Protocol) for communication between vehicles and a fixed proxy in the Internet. MCTP combines several TCP enhancements proposed in section 2. Its core functionality belongs to the category of utilizing information from intermediate systems, which is extended by modifications of the TCP congestion control mechanisms. In general, MCTP is based on the principles of Ad Hoc TCP (ATCP [13]), which relies on information on pending congestions in the network. This idea is combined with an approach similar to TCP Feedback [14] and TCP Stop-and-Go proposed by Ritter [15]. Like ATCP, MCTP implements a sublayer between TCP and IP as depicted in fig. 2. The basic principle of MCTP is that it observes the IP packet flow between sender and receiver in order to react appropriately. Therefore, MCTP considers notifications from underlying protocols as well as from intermediate systems:

- ECN indicates pending congestions detected by intermediate systems.
- Intermediate systems indicate a partitioned network using ICMP destination unreachable messages. This information is relevant for local communication between vehicles only, i.e. for communication without Internet access.
- The mobility management protocol [16] we used is able to notify MCTP in case of disconnections very efficiently.

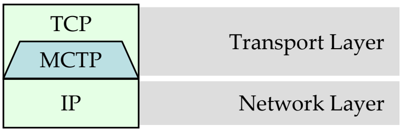


Fig. 2. MCTP in the TCP/IP model

The available information enables MCTP to distinguish between link errors, congestions, network partitions, and disconnections from the Internet. Besides the available information, MCTP also takes into account events caused by TCP itself. Such events are the retransmission timeouts for segments and the arrival of (duplicate) acknowledgements for successfully transmitted segments. Based on this knowledge, MCTP controls the transmission procedure of TCP in different situations by controlling retransmissions and timeouts, and by probing for the network characteristics. MCTP therefore implements its own protocol state machine, which comes into operation after TCP successfully established a connection between the end systems.

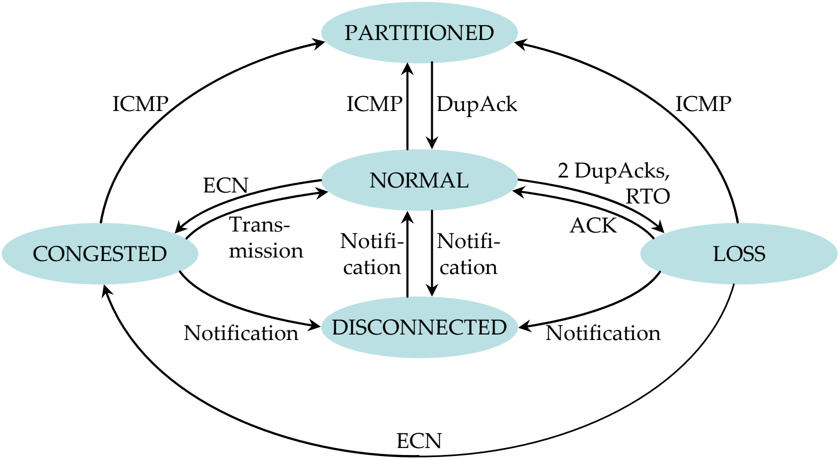


Fig. 3. MCTP protocol state machine

3.1 MCTP Protocol State Machine

A basic feature of MCTP is that it explicitly differentiates between segment losses caused by congestion and segment losses caused by single transmission errors for ongoing connections. MCTP also distinguishes between a partitioned network and a disconnection from the Internet in case of temporary communication breakdowns. A partitioning appears only if a vehicle communicates with another vehicle via multi-hop communication, whereas disconnections occur when a vehicle communicates with a proxy in the Internet. This way, both states can be seen as orthogonal from each other. Fig. 3 shows the protocol state machine. The states NORMAL, LOSS, and CONGESTED are the common operation modes of MCTP in case a data flow is possible. PARTITIONED and DISCONNECTED are only entered when communication is broken.

An important goal of MCTP is to minimize the number of TCP slow starts caused by segment losses. A TCP sender considers a segment as being lost in the following cases:

– receipt of three duplicate acknowledgements (DupAck) for a segment,
– a retransmission timeout (RTO) occurs for a segment.

In the NORMAL state, MCTP counts the number of DupAcks received for a segment. If ECN does not indicate a pending congestion, a segment loss was likely caused by a transmission error. If MCTP receives two DupAcks for a segment in this situation, it enters the LOSS state. Since the TCP congestion control reacts only after the third DupAck, it does not interfere with MCTP in this situation. Similarly, MCTP enters the LOSS state if an RTO expires. In the LOSS state, MCTP forces TCP to freeze its state temporarily. This way, TCP does not invoke congestion control, which would be the wrong thing to do in this situation. Instead, MCTP retransmits the unacknowledged TCP segment. It therefore controls the retransmission timers for the segment accordingly. If an acknowledgement for the segment arrives from the communication peer, MCTP forwards the acknowledgement to TCP, which also recovers TCP, and returns to NORMAL. A different situation occurs when ECN indicates a pending congestion in an intermediate system. Then, MCTP switches to CONGESTED and does nothing: hence, MCTP leaves the congestion control completely to TCP, which handles this situation very efficiently. After the TCP sender transmits a new segment, MCTP returns to NORMAL. This operation mode is similar to ATCP. Differences occur in the handling of DupAcks; whereas ATCP waits for three consecutive DupAcks, MCTP only waits for two DupAcks. Furthermore, MCTP is not based on TCP Reno but uses TCP New Reno with an improved fast retransmit/fast recovery mechanism and selective acknowledgements.

Vehicular mobility may stall ongoing connections in the VANET for a temporary period of time. These communication disruptions are typically caused by a network partitioning or if a gateway becomes unavailable and an alternative gateway cannot be discovered. MCTP considers these two situations and controls TCP appropriately in order to improve the recovery after a connection breakdown. The PARTITIONED state represents a network partitioning that is relevant for inter-vehicle communication only. In contrast, the DISCONNECTED state is entered when the vehicle gets disconnected from the Internet (i.e. the proxy). In case of a network partitioning, an intermediate vehicle will throw an ICMP destination unreachable message if it detects a broken link. If MCTP receives this ICMP message, it moves into the PARTITIONED mode and freezes the current state of TCP. Additionally, it performs a window probing mechanism similar to the zero window probing used in TCP. Thereby, MCTP probes the connection with constant period (the last RTO value). This is in contrast to TCP, which would exponentially backoffs the probing period. If MCTP receives a DupAck from the receiver, the connection is apparently reestablished and communication can be continued. In this case, MCTP recovers TCP, activates the slow start phase of TCP without reducing the slow start threshold, and moves itself back to NORMAL. The PARTITIONED state is also entered from the LOSS state and the CONGESTED state upon receiving an ICMP destination unreachable message. The explicit probing of the connection in case of a network partitioning is optional since it cannot be assumed that a location-based ad hoc routing protocol can detect the reestablishment of the end-to-end routes.

The PARTITIONED mode is of relevance for inter-vehicle communication only. This mode is similar to ATCP; differences between MCTP and ATCP occur in the probing and freezing mechanisms. The PARTITIONED mode is not used when a vehicle communicates with a host in the Internet. In this case, the mobility management protocol is able to detect disconnections very efficiently [17]. If a vehicle looses contact to a gateway, MCTP is notified about the disconnection and switches into the DISCONNECTED mode. In this mode, MCTP completely stops the TCP transmissions and freezes RTO timers. Both TCP and MCTP remain in this state until MCTP is notified about the availability of a new gateway. It then restores TCP and moves itself back to NORMAL. In addition, MCTP activates the slow start phase of TCP without modifying the threshold for the slow start. This allows TCP to converge its data rate to the new situation. Finally, MCTP triggers TCP to retransmit queued segments immediately. If such segments are not available, MCTP sends two acknowledgements in order to generate a DupAck.

4 Evaluation

The goal of the evaluation is to determine the performance of our MCTP Linux implementation together with the communication characteristics of a typical VANET scenario. The VANET communication characteristics were modeled by the NISTNet emulator, which shapes network traffic flows according to configurable parameters like bandwidth, delay, jitter, packet drop rate, and packet duplication rate. Fig. 4 shows our test environment consisting of five connected Linux hosts: on the left-hand side, the mobile node (MN) represents the vehicle that communicates via the proxy (middle) with a correspondent node (CN) in the Internet on the right-hand side. The VANET emulator between MN and

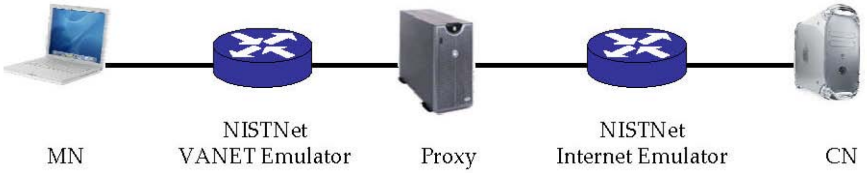


Fig. 4. Test environment used for the evaluation

proxy emulated the communication characteristics a vehicle experienced, and a second emulator between proxy and CN emulated the Internet characteristics.

The communication characteristics in the Internet are highly complex, which make the realistic model almost impossible for the Internet emulator. Instead, we used the following parameters derived from investigations in [18]:

- The bandwidth between proxy and CN is assumed to be higher compared to the bandwidth in the VANET.
- The delay is assumed to be 200 ms with a jitter of ±10 ms.
- The IP packet error rate is 0.2 %. Duplicates are not assumed.

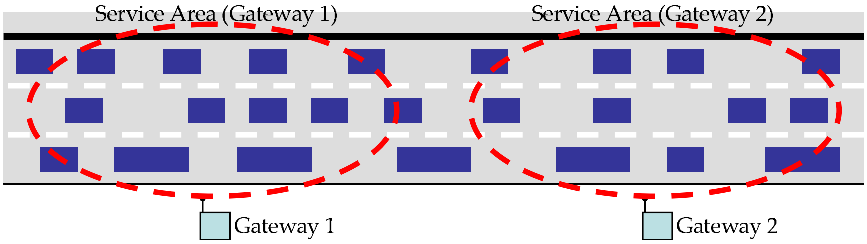


Fig. 5. Highway segment assumed for evaluation

The VANET emulator models a highway segment with a high traffic flow as depicted in fig. 5. Thereby, Internet access is provided by two gateways. The VANET emulator models the communication characteristics a vehicle v experiences while passing this segment. Due to multi-hop communication with an assumed transmission range of 100 m, v is able to communicate with the Internet in the service area (2 km diameter) around each gateway. Fig. 6 shows the "distance" in hops between gateways and v traveling at the right lane. The contact to the first gateway is assumed at 14 hops. v first approaches the gateway resulting in a decrease of the distance every 3 s on average. After 40 s, v enters the direct transmission range of the gateway and contact is lost for a short time after it leaves this range. After 80 s, the first gateway gets unavailable for v and communication is no longer possible for the next 50 s, until v enters the service area of the second gateway. After 215 s, v leaves this service area and communication breaks again. The second scenario assumes a vehicle driving on the left lane at a higher speed resulting in overtaking maneuvers and, thus, a more

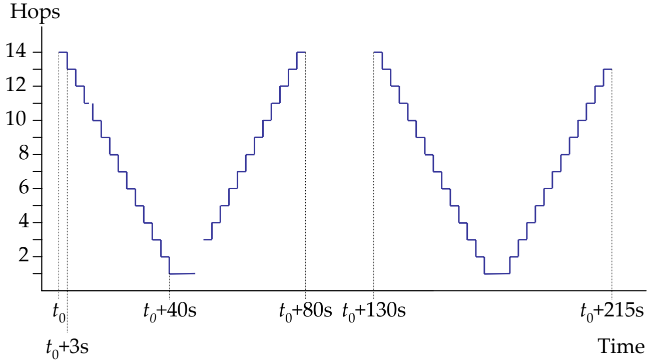
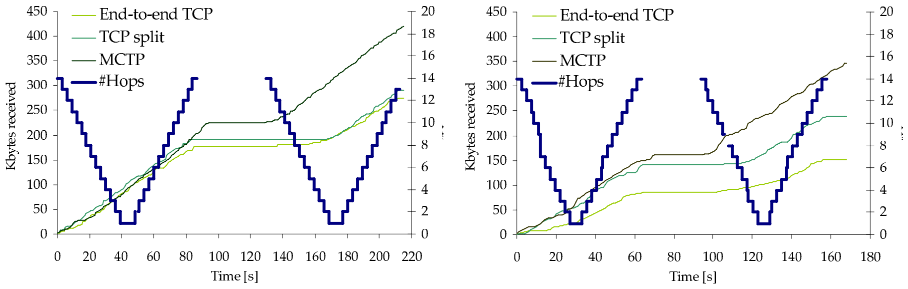


Fig. 6. Distance between vehicle and gateways

unsteady distance graph (cf. fig. 7 (b)). For inter-vehicle communication, we assumed the FleetNet system [1] that has the following characteristics: 588 kbit/s (shared) link layer bandwidth, 40 ms delay, 1 % IP packet error rate per link, and 1 % duplicates, symmetrical communication. On the network layer, we assumed the overhead caused by an optimized gateway discovery protocol for VANETs [16] and a respective mobility protocol for VANETs described in [19]. Thereby, the available bandwidth is shared equally among 27 communicating vehicles, resulting in 21.57 kbit/s on average per vehicle. The path between gateway and proxy was not considered since we assumed an ATM network that connects the gateways to the proxy.

In our test environment, we evaluated three configurations: end-to-end TCP between MN and CN, a proxy that segments the connection into two TCP connections ("TCP split"), and a split proxy using MCTP for communication between MN and proxy. For each configuration, we transferred data from the MN (vehicle) to the CN through both emulators, which reflect the communication characteristics a vehicle experiences in the above scenario. We repeated each measurement three times and took their mean value in order to minimize statistical variations of the NISTNet emulator. Fig. 7 shows the results of the three configurations for the right (a) and left (b) lane. The charts also depict the distance between the vehicle and the gateways to show the correlations. The three graphs in fig. 7 (a) showed similar characteristics in the beginning. This behavior can be expected since decreasing error rates and packet delays typically do not cause slow starts in TCP. The throughput of the three tests decreases slightly when the number of hops increases in the time interval between 50 s and 80 s. This chart also depicts the effects of a longer period of disconnection between 90 s and 130 s: After the reconnection through the second gateway at 130 s, it takes a long time until TCP detects the reconnection and continues with its transmission. Interestingly, end-to-end TCP had a slightly quicker response time, which is explained by statistical deviations of the NISTNet emulator; it took about 35 s until end-to-end TCP and TCP split recovered after the reconnection. The MCTP measurements show a smooth and continuous behavior over the total simulation run. An interesting observation is that MCTP is able to transmit data until the disconnection from the first gateway occurs (at about 90 s) whereas communication in case of end-to-end TCP and TCP split stalled about 10 s before the disconnection from the first gateway occurred. This effect can be explained with the high packet error rates at this distance, which reduces the TCP throughput significantly. After the reconnection to the second gateway at 130 s, MCTP reacts quickly and continues its transmission in the same way than in the beginning of the simulation run. In this phase, the data throughput also increases continuously.



(a) Right lane

(b) Left lane

Fig. 7. Evaluation results

The measurements for the left lane in fig. 7 (b) show that the throughput is lower than on the right lane. This is caused by the shorter connection times to the Internet and the higher variations in the communication characteristics. End-to-end TCP seems to have problems especially in the beginning of the simulation run. It takes about 20 s until end-to-end TCP is able to transmit a noticeable amount of data. This chart also illustrates the problem of TCP with longer periods of disconnections. It takes about 35 s until TCP recovers after the reconnection to the second IGW at about 95 s. In contrast, TCP split has a significantly better performance since the data throughput increases more steadily in the beginning. The TCP split measurement also converge more quickly after the reconnection to the Internet through the second gateway, which takes on average 25 s. The MCTP measurements showed a characteristic similar to the measurements for the right lane. Thereby, the transmission of data segments continues steadily while the vehicle is connected to the Internet. After the reconnection to the second gateway, MCTP reacts quickly and the transmission is continued with a very short delay but suffers from the high packet losses in the beginning.

The measurements showed that MCTP improves communication efficiency at the transport layer in this scenario. MCTP is able to retransmit lost segments very efficiently and, in contrast to TCP, it reacts quickly to disconnections from and reconnections to the Internet and, thus, does not pass up the available bandwidth. In both scenarios, the performance of MCTP is significantly higher compared to the other tests: Over the simulation time, end-to-end TCP transmitted 274.155 Kbyte (left lane: 150.592 Kbyte), TCP split transmitted 291.531 (left lane: 237.955 Kbyte), and MCTP was able to transfer 420.885 Kbyte (left lane: 346.072 Kbyte) of data. Since segment losses and temporary disconnections from the Internet are quite common in vehicular communication scenarios, we can carefully conclude that MCTP is able to improve communication between vehicles and Internet hosts.

5 Conclusion

Communication efficiency is an important issue in vehicular ad hoc networks. In this paper, we propose an optimized transport protocol called MCTP for the Internet access of vehicles through VANETs. MCTP was developed for proxy-based communication architectures where vehicles communicate with a proxy using MCTP, whereas communication between proxy and Internet host is based on standard TCP. MCTP distinguishes different network situations and is, thus, able to control TCP appropriately: MCTP handles segment losses efficiently and reacts to disconnections very quickly. Our evaluation based on an emulated highway segment with a high traffic flow shows that MCTP is able to increase data throughput by a factor of 2.3 compared to traditional end-to-end TCP, and by a factor of 1.5 compared to a split TCP approach. Our evaluation also showed that performance enhancing proxies improve communication performance in vehicular environments.

In our future work, we will examine additional "typical" vehicular communication scenarios. The current status of the MCTP prototype includes the basic protocol mechanisms. We are planning to improve this prototype further on by considering additional available information, e.g. from the routing protocol. This allows us to optimize the slow start phases after disconnections or after a network partitioning. We are also planning additional comparisons with different TCP variants and TCP optimizations. However, most of them are not compatible with our test environment.

References

1. Franz, W., Eberhardt, R., Luckenbach, T.: FleetNet – Internet on the Road. In: Proceedings of the 8th World Congress on Intelligent Transport Systems, Sydney, Australia (2001)
2. Morris, R., Jannotti, J., Kaashoek, F., Li, J., Decouto, D.: CarNet: A Scalable Ad Hoc Wireless Network System. In: Proceedings of the 9th ACM SIGOPS European Workshop, Kolding, Denmark (2000)
3. Bechler, M.: Internet Integration of Vehicular Ad Hoc Networks. Dissertation, Logos-Verlag Berlin, ISBN 3-8325-0750-7 (2004)
4. Bae, S., Xu, K., Lee, S., Gerla, M.: TCP Behavior across Multihop Wireless and Wired Networks. In: Proceedings of the 2002 IEEE GLOBECOM, Taipei, Taiwan (2002)

5. Schiller, J.: Mobile Communications. Addison Wesley (2003)
6. Gerla, M., Sanadidi, M.Y., Wang, R., Zanella, A., Casetti, C., Mascolo, S.: TCP Westwood: Congestion Window Control Using Bandwidth Estimation. In: Proceedings of the 2001 IEEE GLOBECOM, San Antonio, Texas, USA (2001)
7. Wang, F., Zhang, Y.: Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response. In: Proceedings of the 3rd ACM MobiHoc, Lausanne, Switzerland (2002)
8. Sinha, P., Venkitaraman, N., Sivakumar, R., Bharghavan, V.: WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. In: Proceedings of the 5th ACM/IEEE MOBICOM, Seattle, Washington, USA (1999)
9. Fu, Z., Greenstein, B., Meng, X., Lu, S.: Design and Implementation of a TCP-Friendly Transport Protocol for Ad Hoc Wireless Networks. In: Proceedings of the 10th International Conference on Network Protocols (ICNP), Paris, France (2002)
10. Brakmo, L.S., Peterson, L.L.: TCP Vegas: End to End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas in Communications (1995)
11. Sundaresan, K., Anantharaman, V., Hsieh, H.Y., Sivakumar, R.: ATP: A Reliable Transport Protocol for Ad-hoc Networks. In: Proceedings of the 4th ACM MobiHoc, Annapolis, Maryland, USA (2003)
12. Goff, T., Moronski, J., Phatak, D.S., Gupta, V.: Freeze-TCP: A true End-to-End TCP Enhancement Mechanism for Mobile Environments. In: Proceedings of the 19th IEEE Conference on Computer Communications (Infocom), Tel Aviv, Israel (2000)
13. Li, J., Singh, S.: ATCP: TCP for Mobile Ad Hoc Networks. IEEE Journal on Selected Areas in Communications (2001)
14. Chandran, K., Raghunathan, S., Venkatesan, S., Prakash, R.: A Feedback Based Scheme For Improving TCP Performance in Ad-Hoc Wireless Networks. In: Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS), Amsterdam, The Netherlands (1998)
15. Ritter, H.: Bedarfsorientierte Dienstgüteunterstützung durch adaptive Endsysteme. VDI Verlag (2001) [German].
16. Bechler, M., Jaap, S., Wolf, L.: Mobility Management for Vehicular Ad Hoc Networks. In: Proceedings of the 61st IEEE Semiannual Vehicular Technology Conference (VTC 2005 Spring), Stockholm, Sweden (2005)
17. Bechler, M., Franz, W.J., Wolf, L.: Mobile Internet Access in FleetNet. In: Proceedings of the 13th Fachtagung Kommunikation in Verteilten Systemen (KiVS), Leipzig, Germany (2003)
18. Bolot, J.C.: End-to-End Packet Delay and Loss Behavior in the Internet. In: Proceedings of the 1993 ACM SIGCOMM Conference, San Francisco, California, USA (1993)
19. Bechler, M., Storz, O., Franz, W., Wolf, L.: Efficient Discovery of Internet Gateways in Future Vehicular Communication Systems. In: Proceedings of the 57th IEEE Vehicular Technology Conference (VTC), Jeju, Korea (2003)

Cooperative Failure Detection in Overlay Multicast[⋆]

Mengkun Yang and Zongming Fei

Department of Computer Science, University of Kentucky,
301 Rose Street, 2nd floor, Lexington, KY 40506, USA
{myang0, fei}@cs.uky.edu

Abstract. Node failures and ungraceful departures are important issues to be dealt with in overlay multicast. Fast detection is key to minimizing the disruption of service to the affected nodes participating in the multicast session. In this paper, we propose a cooperative failure detection mechanism that can greatly reduce the failure detection time. A significant contribution of the paper is that we quantify three important measures, i.e., the expected detection time, the probability of false failure detection, and the overhead. This allows us to study the fundamental tradeoff among them in the failure detection mechanisms. The analysis and simulations show that the proposed cooperative failure detection mechanism can significantly reduce the failure detection time while maintaining the probability of false positive at the same level, at the cost of slightly increased overhead.

1 Introduction

Overlay multicast (also known as application-layer multicast) [1,2] has been widely investigated as an alternative to IP multicast to implement group communications for its easy deployment. It builds an overlay topology (usually a tree) among end hosts participating in the multicast session, by using the unicast service provided by the substrate network. The duplication functions are implemented at end hosts rather than routers.

Management of the overlay multicast tree faces a key problem. The non-leaf nodes in the tree are end hosts, which are more likely to fail than routers and may leave the multicast tree voluntarily without informing other nodes. In these cases, all of its downstream nodes are partitioned from the multicast tree and cannot get the multicast data any more. It is important to recover from partitioning quickly so that the disruption of service to those downstream nodes is minimized. The time to resume the data flow to those affected nodes is an important measure of the *responsiveness* of failure recovery mechanisms.

The recovery process consists of two steps, *failure detection* and *tree reconstruction*. Failure detection means that when a node in overlay multicast fails or leaves the multicast session, other nodes can detect the event. A departing node may leave the multicast tree gracefully, by sending a message to relevant nodes. The detection is not a problem in this case. However, it is not uncommon in overlay multicast that a node leaves the

[⋆] This work was supported in part by the National Science Foundation under Grants CCR-0204304 and EIA-0101242.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 881–892, 2005.

tree accidentally, such as in case of failures, or leaves voluntarily without informing other nodes about its status. We need a detection mechanism for affected nodes to reach the conclusion that the node is gone as soon as possible. In the rest of the paper, we will use "failure" or "a node fails" to include the case in which a node leaves the multicast tree without informing others.

The tree reconstruction is the process that those orphaned nodes or subtrees find new parents to reconnect to the multicast tree. Several recent researches have addressed the problem. A centralized approach depends on a coordinator to find the locations in the tree for those affected nodes [3]. SpreadIt adopts a distributed approach in which disconnected nodes try to get help from their grandparents or the root of the tree [4]. A proactive approach pre-computes rescue plans before failures happen to reduce the recovery time [5]. They improve the performance of the tree reconstruction process after failures have been detected.

In contrast, failure detection itself has not been widely studied in the context of overlay multicast. They are usually described as a part of the tree construction procedure [1,6], rather than as a focal topic being investigated as it deserves. Lack of quantitative analysis of different detection mechanisms hinders a full exploration of different design options.

In this paper, we perform an analysis of a basic heartbeat mechanism and study the fundamental tradeoff among the failure detection time, the probability of false positive, and the overhead. Based on that, we propose a cooperative approach to node failure detection in overlay multicast to speed up the detection process. The basic idea is that all neighbors interested in the well-being of a node cooperate with each other. To detect the failure of a non-leaf node in overlay multicast, all its children and its parent can form a logical group to help each other to make a decision. When any of them loses a heartbeat, it tells others in the group about this event. If a node cannot receive the heartbeats from the target node, and also receive information from cooperating nodes that heartbeats are missing, it can quickly reach the conclusion with higher confidence that the target node has failed. We perform a formal analysis of the cooperative scheme and design a probabilistic notification technique to deal with big group sizes. Through analyses and simulations, we find that the cooperative scheme can achieve shorter failure detection time and a smaller probability of false positive at the same time, with a small increase of overhead, compared with the basic heartbeat mechanism.

The rest of the paper is structured as follows. Section 2 proposes the cooperative failure detection mechanism. Section 3 gives an analytical study on the gains and costs of different detection schemes. Performance evaluations are presented in Section 4 and related work is discussed in Section 5. We conclude the paper in Section 6.

2 A Cooperative Failure Detection Mechanism

The node failure and its detection can be illustrated by an example in Fig. 2. When non-leaf node 5 fails, all the links (shown as dotted lines) between 5 and other nodes will be affected. All the downstream nodes, 8, 9, 10, and 15-22, are affected and experience data disruptions. Failure detection is the process that neighboring nodes (e.g., nodes 2, 8, 9, and 10) come to a conclusion that the node being monitored (node 5) has failed.

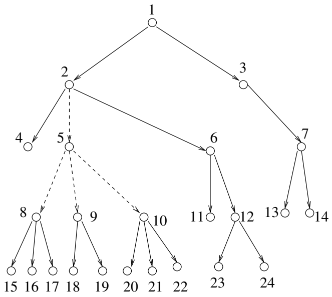


Fig. 1. An example of node failure

In the following discussion, the node being monitored will be called the *target node*. The nodes participating in the failure detection of the target node are usually its *neighbors*, such as nodes $2, 8, 9$, and 10 for target node 5. In the discussion, we also use terms *monitor nodes* and *detecting nodes* interchangeably with term *neighbors*.

2.1 Basic Heartbeat Scheme

The basic heartbeat scheme works as follows. Every node sends a heartbeat message to each of its monitor nodes every $\mathcal{T}$ seconds. If a monitor node does not receive k heartbeat messages in a row, it will derive that the target node has failed. k and $\mathcal{T}$ are design parameters that can be adjusted. They determine the performance of the detection mechanism.

Since each node independently makes its own decision about the failure of other nodes based on its observation on the heartbeat losses, we call this basic heartbeat scheme *non-cooperative* failure detection in our later discussion, in contrast to the cooperative scheme discussed next.

2.2 Cooperative Scheme

The idea of the cooperative scheme is to let monitor nodes of a target node cooperate with each other as a group so that each node can reach the conclusion faster. The goal is that in most cases, a monitor node can detect the failure of a target node within one heartbeat interval, with the help of other nodes in the same group.

Similar to the non-cooperative case, every target node sends a heartbeat message to each of its monitor nodes every $\mathcal{T}$ seconds. When a monitor node does not receive the heartbeat message at the expected time, it will send a notification message to each node in the cooperating group with regard to the same target node. When a monitor node receives the heartbeat message, no notification will be sent. So no extra traffic is generated when the target node works normally.

For each target node, each monitor node maintains two counters in the cooperative scheme. One is the number of heartbeats lost, denoted as N_h, and the other is the number of notifications received, denoted as N_n. They are both initialized to 0.

– When a monitor node receives a heartbeat message from the target node, it resets both counters, N_h and N_n, to 0.

– When it does not receive an expected heartbeat from the target node, it increases the lost heartbeat counter N_h by 1, sends a notification to each node in the group, and performs the following check. If $N_h + N_n \geq k$, it concludes that the target node has failed.
– When it receives a notification from other nodes in the cooperating group with regard to the target node, it increases the notification counter N_n by 1 and performs the following check. If $(N_h + N_n \geq k) \wedge (N_h > 0)$, it concludes that the target node has failed.

Note k is a threshold parameter that can be adjusted. It can be different from the k in the basic heartbeat scheme. A monitor node reaches the conclusion that the target has failed only after it has lost at least one heartbeat from the target node.

3 Analytical Study

To better understand the gains and cost of the failure detection approaches, we perform a formal analysis in this section. Specifically, we want to quantify the following performance measures.

1) *Failure Detection Time.* It is the interval between the time a node fails and the time a monitor node reaches the conclusion that it has failed.
2) *Probability of False Positive.* It is the probability that while a target node works fine, a monitor node reaches the conclusion that it has failed. This may happen when the heartbeat messages are lost.
3) *Overhead.* It is the traffic generated for the failure detection purpose. Specifically, we calculate the average number of messages generated per unit time for *one* monitor node to be able to detect the failure of a target node.

We assume the probability of message loss between any pair of nodes in the overlay multicast tree is p $(0 \leq p \leq 1)$ and losses are independent. For clarity of analysis, we assume the end-to-end path latency between two nodes is negligible and a monitor node knows a heartbeat is lost at the exact same time when the heartbeat is supposed to be sent out. These paths include those not in the overlay tree and yet used in the cooperative failure detection. Every node in the overlay multicast tree fails with probability q $(0 \leq q \leq 1)$. In the cooperative failure detection, we assume the number of nodes in a cooperating group is n. When $n = 1$, there is no other node in the group. The other two important parameters are threshold value k, which determines when a monitor node concludes that a target node has failed, and the heartbeat interval $\mathcal{T}$, which specifies how frequently the heartbeat messages are sent out.

3.1 Failure Detection Time

Non-cooperative Case. In the non-cooperative failure detection, a node can detect the failure of a target node if the number of heartbeats it has missed in a row exceeds the threshold k. Therefore, the failure detection time T_{nc} is between $(k-1)\mathcal{T}$ and $k\mathcal{T}$, as shown in Figure 2. Specifically, $T_{nc} \approx (k-1)\mathcal{T}$ if the target node fails immediately before the expected time at which the next heartbeat should be sent out (point B); and

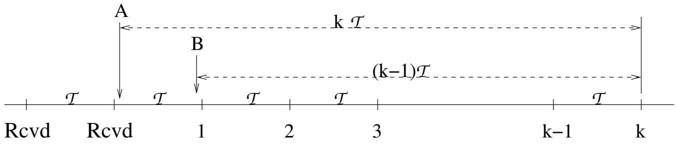


Fig. 2. Failure detection time for the non-cooperative approach

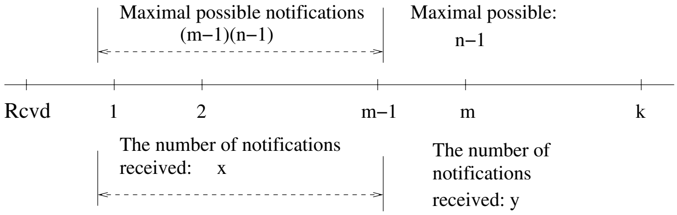


Fig. 3. Failure detection time for the cooperative approach

$T_{nc} \approx k\mathcal{T}$ if the target node fails immediately after a heartbeat is sent out (point A). Assuming that the failure event is uniformly distributed in the interval (between point A and point B), we can get the expected failure detection time for the non-cooperative approach as $E(T_{nc}) = \left(k - \frac{1}{2}\right)\mathcal{T}$.

Cooperative Case. In the cooperative failure detection, the longest failure detection time is $T_{co} = k\mathcal{T}$, when the target node fails immediately after a heartbeat is sent out and the detecting node gets no notifications from other nodes in the group. The shortest failure detection time is achieved when the target node fails immediately before a heartbeat is supposed to be sent out, and the detecting node gets all notifications from all other nodes in the cooperating group. For every scheduled heartbeat, the detecting node gets $n-1$ notifications and one missing heartbeat. It takes $\lceil \frac{k}{n} \rceil$ missing heartbeats to get the total bigger than or equal to the threshold k. So the minimal detection time is $T_{co} = \left(\lceil \frac{k}{n} \rceil - 1\right) \times \mathcal{T}$.

We are interested in the expected detection time. If the probability the detecting node can reach the conclusion after m ($\lceil \frac{k}{n} \rceil \leq m \leq k$) missing heartbeats is $g(m)$, the expected detection time will be

$$E(T_{co}) = \mathcal{T} \sum_{m=\lceil k/n \rceil}^{k} mg(m) - \frac{\mathcal{T}}{2} \tag{1}$$

To get the probability $g(m)$, we explore all possible cases that a detecting node cannot make the conclusion in $m-1$ intervals, but can in m intervals. Assume the number of notifications received in the first $m-1$ intervals is x, and the number of notifications received in the m-th interval is y, as shown in Fig. 3. Because the maximum number of notifications that can be received in one interval is $n-1$, we have $0 \leq x \leq (n-1)(m-1)$ and $0 \leq y \leq n-1$. The fact that the detecting node cannot make the conclusion in the first $m-1$ intervals implies $x+m-1 \leq k-1$, i.e., $x \leq k-m$. In order for the node to make the conclusion in the m-th interval, we have $x+y+m \geq k$.

From $y \leq n-1$ and $x+y+m \geq k$, we get $x \geq k-m-(n-1)$. Therefore, x can take values from $max(0, k-m-(n-1))$ to $min(k-m, (n-1)(m-1))$, and y can take values from $k-m-x$ to $n-1$.

Because the target node has failed, all other nodes in a cooperating group will send a notification to the detecting node. Each notification will reach the detecting node with probability $1-p$. Under the independent loss assumption, the number of notifications reaching the detecting node in the first $m-1$ intervals follows the binomial distribution with parameters $((n-1)(m-1), 1-p)$. The number of notifications reaching the detecting node in the m-th interval follows the binomial distribution with parameters $(n-1, 1-p)$. We use notation $P_{n,p}(i)$ to represent the probability of i successes in the binomial distribution with parameters (n,p). We know $P_{n,p}(i) = \begin{pmatrix} n \\ i \end{pmatrix} p^i (1-p)^{n-i}$, where $\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}$.

The probability of getting x notifications in the first $m-1$ intervals is $P_{(n-1)(m-1),1-p}(x)$. The probability of getting y notifications in the m-th interval is $P_{n-1,1-p}(y)$. Therefore, if we let $\alpha = (n-1)(m-1)$, and $\beta = k-m-(n-1)$, the probability of reaching the conclusion in the m-th interval is

$$g(m) = \sum_{x=max(0,\beta)}^{min(k-m,\alpha)} \left(P_{\alpha,1-p}(x) \sum_{y=k-m-x}^{n-1} P_{n-1,1-p}(y) \right)$$

Combining this with formula (1), we get the expected detection time of the cooperative approach.

3.2 Probability of False Positive

Non-cooperative Case. In the non-cooperative failure detection, a false failure detection at node x is caused by the loss of k consecutive heartbeats that the target node sends to x. Therefore, the probability of false positive is $F_{nc} = p^k$.

Cooperative Case. In the cooperative detection, both the consecutive loss of heartbeats sent to a monitor node v itself and the notifications from cooperating nodes contribute to the false failure detection. The probability that a cooperating node will send a notification to v while the target node is still sending heartbeats is $\theta = p(1-p)$, which is the probability (p) that the heartbeat to that node is lost times the probability $(1-p)$ that the notification successfully reaches v.

The probability that v concludes the target node has failed after the target node sends out m heartbeat messages is the probability that v misses all m heartbeat messages times the probability that an appropriate number of notifications reach v. Note the number of notifications reaching v in the first $m-1$ intervals follows the binomial distribution with parameters (α, θ), and the number of notifications reaching v in the m-th interval follows the binomial distribution with parameters $(n-1, \theta)$. Following the similar reasoning as in deriving the expected time, the probability of false positive when node v reaches a conclusion that the target node has failed after m ($\lceil k/n \rceil \leq m \leq k$) heartbeat messages is

$$f_{co}(m) = p^m \sum_{x=\max(0,\beta)}^{\min(k-m,\alpha)} \left(P_{\alpha,\theta}(x) \sum_{y=k-m-x}^{n-1} P_{n-1,\theta}(y) \right) \tag{2}$$

where $\alpha = (n-1)(m-1)$, and $\beta = k - m - (n-1)$. Therefore, the total probability of false positive in the cooperative case is $F_{co} = \sum_{m=\lceil k/n \rceil}^{k} f_{co}(m)$.

3.3 Overhead

Non-cooperative Case. The overhead for one monitor node is one heartbeat message per $\mathcal{T}$ seconds, if the target node does not fail. Therefore, the overhead is $H_{nc} = \frac{1-q}{\mathcal{T}}$.

Cooperative Case. For the cooperative failure detection, in addition to the heartbeat message, we have notification messages transmitted between cooperating nodes. When the target node fails (with probability q), each monitor node will lead to $n-1$ notifications being transmitted. When the target node sends heartbeats normally (with probability $1-q$), each of $n-1$ cooperating nodes may miss the heartbeat message with probability p and send a notification message to the monitor node. Therefore, the overhead for the cooperative case is

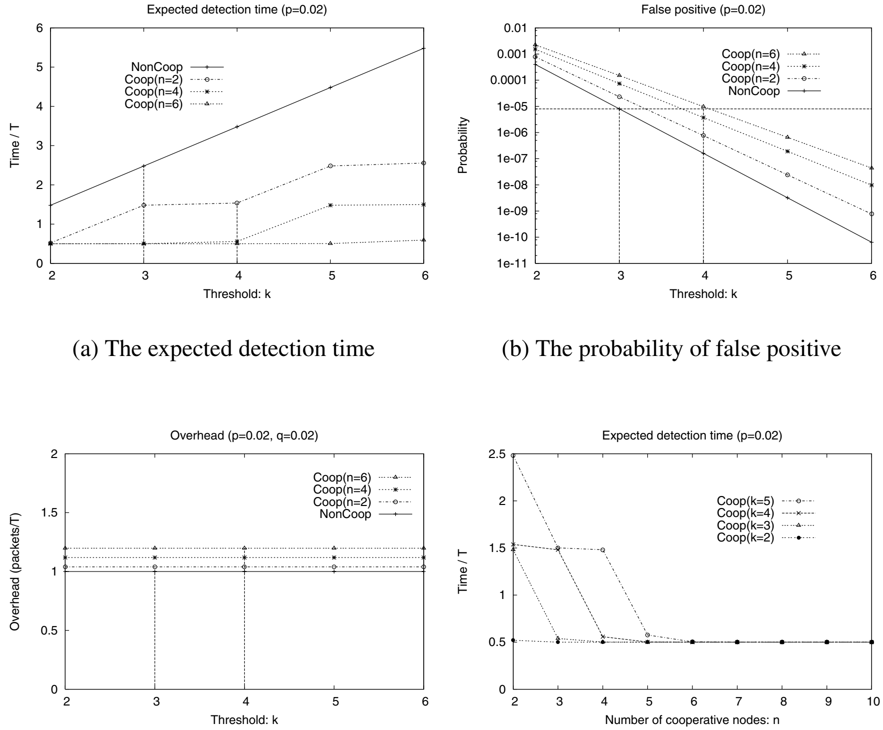
$$H_{co} = \frac{q(n-1) + (1-q)\left(1 + p(n-1)\right)}{\mathcal{T}} \tag{3}$$

3.4 Numerical Results and Analysis

Non-cooperative Versus Cooperative. To compare different approaches, we show the numerical results in Fig. 4(a)-4(c). For the cooperative detection, the number of nodes in a cooperative group (n) is either 2, 4 or 6. As the threshold k increases, the detection time increases in all schemes (Fig. 4(a)), but the non-cooperative approach increases much faster than the cooperative approach. For a given threshold k, the cooperative approach can detect the failure in significantly shorter time. For example, when $k = 4$, the cooperative approach with $n = 6$ only takes $0.5\mathcal{T}$, which is 1/7 of the time ($3.5\mathcal{T}$) by the non-cooperative approach. Fig. 4(b) and Fig. 4(c) show that the non-cooperative approach has a lower probability of false positive and lower overhead than the cooperative approach, when they use the same threshold value (k).

An interesting observation is that the cooperative scheme with $n = 4$ nodes in a group and the threshold $k = 4$ achieves *both* a lower probability of false positive and the shorter detection time than the non-cooperative approach with $k = 3$. A more general observation from the plot is that given a non-cooperative scheme with some $k > 1$, we can always find a cooperative scheme with appropriate k and n such that it has shorter detection time and a lower probability of false positive at the same time. In all cases, the extra overhead is always no bigger than 20%.

The Effect of the Number of Monitor Nodes in a Cooperative Group. In Fig 4(d), as the number of monitor nodes in a group increases from 2 to 10, the detection time decreases. Due to lack of space, we do not show the figures for the probability of false positive and the overhead. Both metrics increases as the size of the cooperative group increases. The key observation is that when the group size increases beyond threshold



(a) The expected detection time

(b) The probability of false positive

(c) The detection overhead

(d) The effect of group size on detection time

Fig. 4. Quantitative comparison of failure detection schemes

k, the detection time reaches the minimum value $\mathcal{T}/2$ and cannot be further decreased, while the probability of false positive and the overhead continue increasing. On one hand, this tells us that the cooperative scheme with $n \geq k$ achieves the design goal of having monitor nodes detect the failure of a target node within one heartbeat interval in most cases. On the other hand, this gives us a hint that the number of nodes in a group should not be too large. A value close to or a little bit bigger than k is enough.

An Implementation Implication. When implementing the cooperative scheme, we let the parent and the children of a target node form a cooperative group, which can be much larger than k. The implication of the above observation is that the large number of members in a group does not help much to reduce the detection time further, but increases the traffic generated when a heartbeat is lost. One approach to solving the problem is to limit the number of cooperating nodes in a group to k. In this paper, we adopt a different approach, i.e., probabilistic notification. We can let a node send notifications with probability p_c ($0 \leq p_c \leq 1$) if the number of members in a group is too large. Assume the number of nodes in the group is n $(n > k)$. We can set p_c to about $\frac{k-1}{n-1}$, so that each monitor node can roughly receive $k-1$ notifications within one interval and detect the failure if there is one. Also the overall traffic is reduced. We call this method the *probabilistic notification* scheme in the performance evaluation. In contrast, the original method will be called *non-probabilistic notification* scheme.

4 Performance Evaluations

1) Simulation Setup

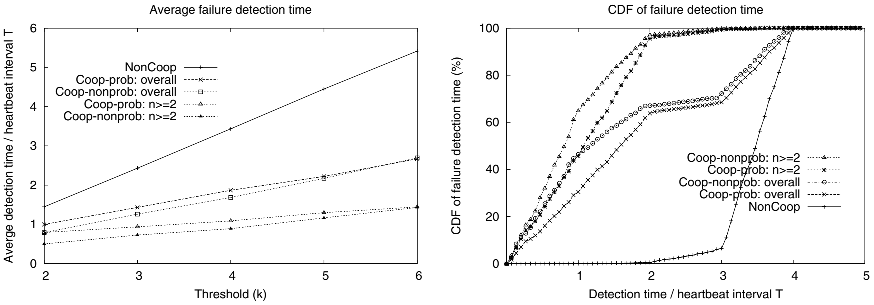
In the simulation, we use GT-ITM [7] to generate 1600 node transit-stub topologies as the underlying network. The source and the multicast members are randomly distributed in stub domains. The network-layer link latencies, represented by the edge weights in the graph, range from 1 ms to 81 ms. The application-level distance (path latency) between two end-hosts is the sum of link latencies on the shortest path between them. It ranges from 1 ms to 220 ms with the average equal to 96 ms in the generated topology. The maximum number of neighbors that a node can have in the overlay multicast tree, called node degree, is uniformly distributed in range $[2, 2\times d-2]$, where d is the average node degree. In the simulations, $d=5$ if not stated otherwise.

All experiments begin with a multicast tree with 160 end-hosts. Then nodes join and leave the tree dynamically, following the Poisson process with leaving and join rate $\lambda = 0.2/second$. Each experiment lasts for two hours. We use the Gilbert model [8] to simulate the packet loss on the network-layer links. The average link loss rate is set as $p_{link} = 1\%$ if not mentioned otherwise. The end-to-end path loss probability is $p = 1-(1-p_{link})^l$ if the path consists of l links. The average number of links between neighbors in the generated multicast tree is about 6. In the experiments, the heartbeat interval is $\mathcal{T} = 15$ seconds.

In the following figures, labels "Coop-nonprob" and "Coop-prob" denote the cooperative failure detection using the non-probabilistic and the probabilistic notification scheme respectively. "NonCoop" represents the non-cooperative approach.

2) Failure Detection Time

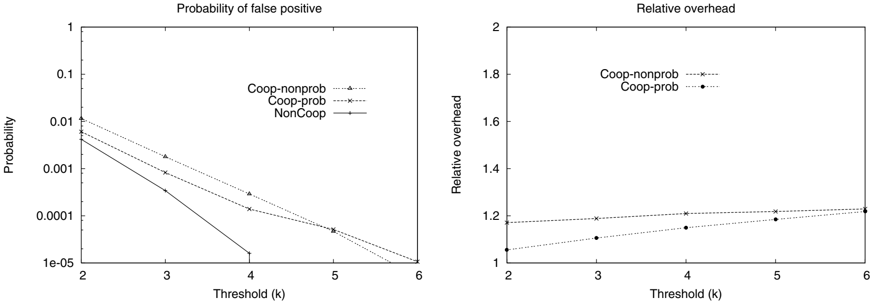
Fig. 5(a) compares the average failure detection time, measured in the number of heartbeat intervals. Fig. 5(b) depicts the cumulative distribution of the failure detection time when the threshold is $k=4$. Curves "overall" plot the average detection time of all nodes in the multicast tree, while curves "$n \geq 2$" only consider those nodes that have $n-1$ nodes to cooperate with in monitoring target nodes. We observe that the cooperative detection achieves much smaller average detection time, compared with the non-cooperative approach. For example, for the nodes in the cooperating groups with sizes $n \geq 2$, when $k=4$, the average detection time is only 30% of the non-cooperative approach. Moreover, more than 55% of them detect the target node failures within just one interval and more than 95% of them detect the failures within two intervals. In contrast, using the non-cooperative approach, more than 95% of the failures are detected after three intervals. The detection time of the "overall" curves are not as good as "$n \geq 2$" because the nodes that have nobody to cooperate with need longer time to detect the failures. We vary the average node degree d between 2.5 and 5 and find that about $70\% \sim 85\%$ of the nodes in the multicast tree belong to the monitoring groups with sizes no less than 2. This means that most of the tree nodes cooperate with



(a) Average failure detection time

(b) CDF of the failure detection time

Fig. 5. Failure detection time



(a) Probability of false positive

(b) Relative overhead

Fig. 6. Probability of false positive and overhead

somebody else and thus can benefit from the cooperative approach to detect the target failures faster. Also we observe that in the cooperative detection, using the probabilistic notification can detect failures almost as fast as using the non-probabilistic scheme.

3) Probability of False Positive

Fig. 6(a) plots the probability of false positive. Given a threshold k, the cooperative approach has more false positives than the non-cooperative approach. However, we also note that using a larger threshold k in the cooperative approach (e.g., $k = 4$) can result in the smaller probability of false positive and the smaller detection time This revalidates the observation in the formal analysis. Moreover, when the threshold k is less than the sizes of most cooperating groups, the probabilistic notification scheme can get smaller probability of false positive for the cooperative approach, compared with the non-probabilistic scheme. This is because using the probabilistic notification, when the cooperating group size n is greater than the threshold k, only about $k - 1$ instead of $n - 1$ cooperating nodes of a monitor node can send false notifications to it. A related problem is how to choose k. We may have a target probability of false positive we want to achieve. Then we can determine the value for k that satisfies the goal.

4) Overhead
The overhead of detection approaches is measured by the number of control packets sent per second for the detection purpose. In the non-cooperative approach, the heartbeat packets are counted. For the cooperative detection, both the heartbeats and the notification packets are included in the calculation. We define the *relative overhead* of the cooperative approach as the ratio of its overhead to that of the non-cooperative approach, under the same experiment configuration.

Fig. 6(b) depicts the relative overhead of the cooperative failure detection using two different notification schemes. We observe that using the non-probabilistic notification, the cooperative approach introduces about 20% more traffic than the non-cooperative approach. This allows us to reduce the average detection time for all tree nodes by 50% and for nodes having somebody to cooperate with by roughly $70 \sim 75\%$, compared with the non-cooperative approach. Moreover, the overhead can be further reduced by using the probabilistic notification scheme. As demonstrated in the figure, its overhead is always less than or equal to that of the non-probabilistic scheme. For example, with $k = 2$, the probabilistic notification only gives rise to 5% more overhead than the non-cooperative approach, in contrast to 17% in the non-probabilistic scheme.

5 Related Work

Failure detection has been studied in distributed systems [9]. The goal is to design a scalable detection mechanism that all nodes in the distributed system can reach a consensus that a failure has occurred. In the context of overlay multicast, failure detection is usually described as a part of the tree construction procedure. In the End System Multicast [1], each node sends *refresh* (heartbeat) messages to its neighbors. If a node does not receive refreshments from a neighbor for a long period of time, a probe is sent to this neighbor. It is considered to be dead if no response is returned. For those nodes it does not receive refreshments for some time, but not long enough, it probabilistically probes them. In the Intradomain Overlays [10], each overlay node periodically sends *keepalive* (heartbeat) messages to its tree parent. Failure in receiving such a message makes the parent to probe the child. No response for the probe leads to declaration of child failure.

Both of them use the heartbeat mechanism to detect the failure. They also use probe to make sure that the target has failed, in order to reduce the probability of false positive. Notice that the final probe itself may also have a certain probability of false positive, depending on which probing method is used. While the probing can also be used in our systems, the cooperative scheme we proposed can reduce the time the monitor node detects the problem at the target node and increase the confidence of the conclusion. By choosing an appropriate threshold and forming a cooperating group, we can decrease the probability of false positive to a very low level and make the probing unnecessary.

The Resilient Overlay Network [11] also addressed the failure detection problem. It aims at detecting path failures instead of node failures in overlay networks, by using an active probing mechanism. If no response is returned after probing a node, a higher probing frequency replaces the normal probing frequency. If no response is received for a certain number of consecutive probings, the probed path is determined to be dead. The traffic generated by the probing process doubles that of heartbeat mechanism, and the chance of message loss is also doubled. Therefore, the probability of false positive is higher than one-way heartbeat based detection mechanisms. Based on the paper, the detection time of the probing mechanism is about $2\mathcal{T}$ if the probing interval is $\mathcal{T}$. In contrast, our cooperative scheme can detect the failure within one heartbeat interval if we have enough cooperative nodes.

6 Concluding Remarks

Failure detection and recovery is a very important problem in overlay multicast. The effectiveness of the detection mechanism has direct impacts on the service quality to the receivers in the session. In this paper, we proposed a cooperative approach to speed up the failure detection process. We gave a quantitative study of three important performance measures of detection mechanisms, and analyzed the fundamental tradeoff among them. We also discussed some implementation issues and evaluated the performance of the proposed scheme. While the focus of this paper is on failure detection in overlay multicast, the cooperative detection mechanism is also applicable to other overlay networks as well.

References

1. Chu, Y.H., Rao, S.G., Seshan, S., Zhang, H.: A case for end system multicast. In: Proceedings of ACM SIGMETRICS'00. (2000) Santa Clara, CA.
2. Chawathe, Y., McCanne, S., Brewer, E.A.: An architecture for Internet content distribution as an infrastructure service (2000) Available at http://www.cs.berkeley.edu/ yatin/papers/scattercast.ps.
3. Padmannabhan, V., Wang, H., Chou, P.: Resilient peer-to-peer streaming. In: Proceedings of the 11th IEEE ICNP'03. (2003)
4. Deshpande, H., Bawa, M., Garcia-Molina, H.: Streaming live media over a peer-to-peer network (2001) Technical Report CS-2001-31, CS Dept. Stanford University.
5. Yang, M., Fei, Z.: A proactive approach to reconstructing overlay multicast trees. In: Proceedings of the IEEE INFOCOM'04. (2004)
6. Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, M.F., James W. O'Toole, J.: Overcast: Reliable multicasting with an overlay network. In: Proceedings of the 4th OSDI'00. (2000)
7. Zegura, E.W., Calvert, K., Bhattacharjee, S.: How to model an internetwork. In: Proceedings of INFOCOM'96. (1996)
8. Yajnik, M., Moon, S., Kurose, J., Towsley, D.: Measurement and modelling of the temporal dependence in packet loss. In: Proceedings of INFOCOM'99. (1999) New York.
9. van Renesse, R., Minsky, Y., Hayden, M.: A gossip-style failure detection service. In: Proceedings of Middleware'98. (1998) 55–70 The Lake District, England.
10. Kommareddy, C., Guven, T., Bhattacharjee, B., La, R., Shayman, M.: Intradomain overlays: Architecture and applications. Technical Report UMIACS-TR 2003-70, University of Maryland, College Park (2003)
11. Anderson, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay netorks. In: Proceedings of ACM SOSP'01. (2001)

Eliminating Bottlenecks in Overlay Multicast*

Min Sik Kim, Yi Li, and Simon S. Lam

Department of Computer Sciences,
The University of Texas at Austin
{minskim, ylee, lam}@cs.utexas.edu

Abstract. Recently many overlay multicast systems have been proposed due to the limited availability of IP multicast. Because they perform multicast forwarding without support from routers, data may be delivered multiple times over the same physical link, causing a bottleneck. This problem is more serious for applications demanding high bandwidth such as multimedia distribution. Although such bottlenecks can be removed by changing overlay topology, a naïve approach may create bottlenecks in other parts of the network. In this paper, We propose an algorithm that removes all bottlenecks caused by the redundant data delivery of overlay multicast, detecting such bottlenecks using a wavelet-based technique we recently proposed. In a case where the source rate is constant and the available bandwidth of each link is not less than the source rate, our algorithm guarantees that every node receives at the full source rate. Simulation results show that even in a network with a dense receiver population, our algorithm finds a tree that satisfies all the receiving nodes while other heuristic-based approaches often fail.

1 Introduction

Recently, many overlay multicast systems have been proposed as alternatives to IP multicast. Overlay multicast systems provide more flexibility in topology construction, but consume more bandwidth of the underlying network because data is often delivered multiple times over the same physical link, causing a bottleneck. This problem is more serious for applications demanding high bandwidth such as multimedia distribution. One way to mitigate the problem is to limit the fan-out of internal nodes in a multicast tree [1]. However, deciding the right number of children is non-trivial; fan-out should be a function of the available bandwidth to each child and the network topology. Furthermore, a bottleneck may be caused by flows from different source (parent) nodes. In such a case, fan-out has little to do with the bottleneck. Therefore, a better way to avoid bottlenecks is to identify them by finding unicast flows in the multicast session that traverse those bottlenecks, and remove them by changing the multicast tree topology. We recently presented a wavelet-based technique, called DCW (Delay Correlation with Wavelet denoising), which can be used to detect flows sharing

* Research sponsored by National Science Foundation ANI-0319168 and CNS-0434515.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 893–905, 2005.

bottlenecks (or congested links) [2]. While other techniques require that flows should share a common end point [3, 4, 5], DCW can be used for any pair of Internet flows with different sources and different sinks.

Identified shared bottlenecks should be eliminated by changing the overlay tree topology. However, it should be done very carefully. When a tree edge is cut, another edge must be added to maintain connectivity. But the newly added edge may cause another bottleneck. Even worse, eliminating the new bottleneck may reincarnate the old one, resulting in oscillation. The algorithm we propose in this paper removes shared bottlenecks without incurring such oscillation. We prove that the algorithm always terminates, and that on termination there is no shared bottleneck in the multicast tree. In a case where the source rate is constant and the available bandwidth of each link is not less than the source rate, our algorithm guarantees that every node receives at the full source rate. We implement our algorithm in a distributed fashion, and compare its performance with other heuristically-built multicast trees. Simulation results show that even in a network with a dense receiver population, our algorithm finds a tree that satisfies all the receiving nodes while other heuristic-based approaches often fail.

The remainder of this paper is organized as follows. Section 2 presents our network model. Section 3 proposes an algorithm to eliminate shared bottlenecks, and Sect. 4 sketches the implementation of a distributed protocol based on the proposed algorithm. Section 5 shows experimental results, and we conclude in Sect. 6.

2 Model

Our network model consists of two layers. The lower layer represents the underlying traditional network with links and nodes, where routing between nodes is done through the lowest-cost path. The upper layer is an overlay network, where a subset of the nodes in the lower layer form a multicast tree.

2.1 Underlying Network

An underlying network is given as a directed graph $G = (N, L)$, where N is a set of nodes in the network, and L is a set of unidirectional links between two nodes in N. Each link $(m, n) \in L$ has two properties: $B(m, n)$, the bandwidth of the link available to overlay multicast, and $c(m, n)$, the cost of the link. The cost is a positive constant and used as a routing metric to compute shortest paths. We assume symmetric routing, i.e. $c(m, n) = c(n, m)$.

Given two nodes u and v in N, the shortest path between them is specified as a set of links $P_L(u, v) = \{(u, n_1), (n_1, n_2), \ldots, (n_i, v)\}$ chosen to minimize the total cost of the links in the set. If there are more than one such paths, we assume that the routing algorithm always selects the same path among them.

2.2 Overlay Multicast Tree

A multicast tree is built on top of the underlying network G, using a set of end-hosts H. H is a subset of N, and consists of end-hosts participating the multicast session. The multicast tree is represented as a set $T = \{(u, v)|u, v \in H,\ v \text{ is a child of } u \text{ in the tree}\}$. We call each element of T an edge of the tree.

Similarly to P_L, P_T is defined as a path in a multicast tree T. Formally, $P_T(u, v) = \{(u, h_1), (h_1, h_2), \ldots, (h_i, v)\}$, where $P_T(u, v) \subset T$.

2.3 Bottleneck

We model multicast traffic as a set of flows; every edge of an overlay multicast tree has an associated flow for data delivery. Each flow f has a source node $\mathit{Src}(f)$, a sink node $\mathit{Snk}(f)$, and the rate of the flow $\mathit{Rate}(f)$.

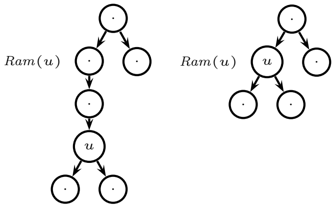
Let $F(m, n)$ be a set of flows passing through the link $(m, n) \in L$. Formally, $F(m, n) = \{f|(m, n) \in P_L(\mathit{Src}(f), \mathit{Snk}(f))\}$. A link (m, n) is a *bottleneck* of the multicast session if and only if $B(m, n) < \sum_{f \in F(m,n)} \mathit{Rate}(f)$. The bottleneck link (m, n) is also called a *shared bottleneck* if multiple flows are passing through the link, or $|F(m, n)| > 1$, where the notation $|S|$ denotes the number of elements of a set (or vector) S.

3 Algorithm

The goal of our algorithm is to remove shared bottlenecks in a multicast tree, so that they cannot throttle throughput. In the algorithm we assume that each bottleneck shared by multiple flows can be detected accurately using a technique such as DCW [2]. Before we describe the algorithm, we define notation to be used in explanation.

- $r \in H$ denotes the root node of a multicast tree.
- $d(u, v)$ is the distance between u and v on T, namely $d(u, v) = |P_T(u, v)|$.
- $\mathit{Parent}(u)$ is the parent node of u in the tree.
- $\mathit{SLeaf}(u)$ is one of the shallowest leaves in a subtree rooted at u. In other words, $\mathit{SLeaf}(u)$ is a leaf node closest to u in the subtree.
- $\mathit{Ram}(u)$ is the node that has caused ramification of the branch of u in the tree, or $\varnothing$ if there is no such node. Formally, $\mathit{Ram}(u)$ is a node along the path from r to u such that $\mathit{Parent}(\mathit{Ram}(u))$ has more than one child, and all the nodes between $\mathit{Ram}(u)$ and $\mathit{Parent}(u)$, inclusively, have only one child. See Fig. 1.

Shared bottlenecks need to be treated differently depending on their relative locations in the tree. There are two types of shared bottlenecks: intra-path and inter-path shared bottlenecks, as shown in Fig. 2, where thick arrows represent flows sharing the same bottleneck. Suppose that a link $(m, n) \in L$ is a shared bottleneck. Then there exist two edges (u_1, v_1) and (u_2, v_2) such that $(u_1, v_1), (u_2, v_2) \in T$ and $(m, n) \in P_L(u_1, v_1) \cap P_L(u_2, v_2)$. Without loss of generality, we assume $d(r, u_1) \leq d(r, u_2)$. A shared bottleneck (m, n) is called an



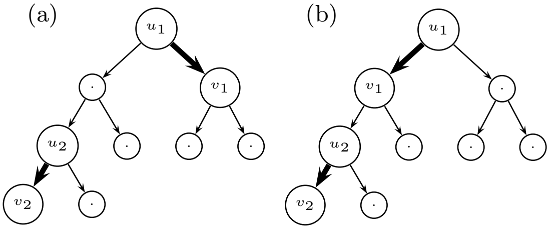


Fig. 1. Ramification point **Fig. 2.** (a) Inter- and (b) intra-path shared bottlenecks

intra-path shared bottleneck of (u_1, v_1) and (u_2, v_2) if $(u_1, v_1) \in P_T(r, u_2)$, and otherwise an *inter-path shared bottleneck*. In this section, we describe first the algorithm for the more general case, inter-path shared bottlenecks, and then the algorithm for intra-path shared bottlenecks. By applying these algorithms iteratively, we can remove all shared bottlenecks in finite iterations. To make sure that it terminates, we will prove that the tree after each iteration is different from any tree in previous iterations.

For proof, we define two properties of a multicast tree: the leaf distance vector and total cost. The leaf distance vector is defined as $D = (d(r, u_1), d(r, u_2), \ldots, d(r, u_k))$, where $u_1, u_2, \ldots, u_k$ are all the leaf nodes in T, and $d(r, u_i) \leq d(r, u_{i+1})$ for every $i < k$. Distance vectors are ordered as follows. For two distance vectors, D and D', D precedes D' ($D \prec D'$) if and only if (i) $|D| > |D'|$, or (ii) $|D| = |D'|$ and D precedes D' in lexicographical order.

The second property, total cost C, is defined to be the sum of costs of all edges in the tree, where the cost of an edge is the sum of all link costs along the edge. Formally, $C = \sum_{(u,v) \in T} \sum_{(m,n) \in P_L(u,v)} c(m, n)$. For each link shared by multiple edges, its link cost is counted multiple times.

3.1 Inter-path Shared Bottleneck

The algorithm to remove an inter-path shared bottleneck is shown in Fig. 3. See also Fig. 2(a) for illustration. When an inter-path shared bottleneck is detected between two edges (u_1, v_1) and (u_2, v_2), the edge farther from the root, (u_2, v_2), is removed and the detached subtree rooted at v_2 is moved to the subtree rooted at v_1. If the shallowest leaf in v_1's subtree is not deeper than v_2, then it is chosen as the node to which v_2's subtree is attached. In this way, we can avoid increasing the fan-out of an internal node, which may affect flows from the node to the existing child nodes. However, since we do not want the tree to become too tall, we also avoid attaching v_2 to a very deep node. Therefore, if the shallowest leaf of v_1 is deeper than v_2, v_2 is attached to a node on the path from v_1 to its shallowest leaf such that the depth of v_2 increases at most by one.

If u_2 becomes a leaf after removing (u_2, v_2), we relocate its branch (the path from $Ram(u_2)$ to u_2) under another leaf in Lines 8–10, because leaving behind u_2's branch may cause oscillation. Suppose that the edge added to connect v_2 to the tree causes another shared bottleneck. Then it is possible that v_2 is detached once again and moved back to u_2, if u_2 is the chosen shallowest leaf in this case.

```
REMOVE-INTER-PATH-SHARED-BOTTLENECK
▷ $(u_1, v_1)$ and $(u_2, v_2)$ in $T$ are sharing a bottleneck, and $d(r, u_1) \leq d(r, u_2)$.
if $d(r, SLeaf(v_1)) \leq d(r, v_2)$
    $T \leftarrow T \cup \{SLeaf(v_1), v_2\} - \{(u_2, v_2)\}$
else
    $t \leftarrow$ a node such that $d(r,t) = d(r, v_2)$ and $\exists P_T(u_1, t) \neq \varnothing$, $P_T(u_1, t) \subset P_T(u_1, SLeaf(v_1))$
    $T \leftarrow T \cup \{(t, v_2)\} - \{(u_2, v_2)\}$
if $\{(u_2, x) | (u_2, x) \in T\} = \varnothing$
    $u, v \leftarrow Parent(Ram(u_2)), Ram(u_2)$
    $c \leftarrow \arg\min_{i \in \{w | Parent(w) = u, w \neq v\}} d(i, SLeaf(i))$
    $T \leftarrow T \cup \{(SLeaf(c), v)\} - \{(u, v)\}$
```

Fig. 3. Removal of an inter-path shared bottleneck

Thus the change made to remove the shared bottleneck between (u_1, v_1) and (u_2, v_2) is reverted, and it revives the bottleneck that we removed earlier. By relocating u_2's branch when u_2 becomes a leaf, we can avoid such oscillations. The following lemma states that the leaf distance vector before REMOVE-INTER-PATH-SHARED-BOTTLENECK algorithm always precedes that after REMOVE-INTER-PATH-SHARED-BOTTLENECK. The proof is in [6].

Lemma 1. *Let D and D' denote leaf distance vectors before and after* REMOVE-INTER-PATH-SHARED-BOTTLENECK *respectively. Then we have $D \prec D'$.*

3.2 Intra-path Shared Bottleneck

Figure 4 presents the algorithm to remove an intra-path shared bottleneck. See also Fig. 2(b) for illustration. Some intra-path shared bottlenecks may be treated like inter-path shared bottlenecks, but others should be treated differently.

```
REMOVE-INTRA-PATH-SHARED-BOTTLENECK
▷ $(u_1, v_1)$ and $(u_2, v_2)$ in $T$ are sharing a bottleneck, and $d(r, u_1) \leq d(r, u_2)$.
if $SLeaf(v_1) \neq SLeaf(v_2)$
    REMOVE-INTER-PATH-SHARED-BOTTLENECK
else if $Ram(v_1) \neq Ram(v_2)$
    $u, v \leftarrow Parent(Ram(v_2)), Ram(v_2)$
    $c \leftarrow \arg\min_{i \in \{w | Parent(w) = u, w \neq v\}} d(i, SLeaf(i))$
    if $d(u, SLeaf(c)) \leq d(r, u) + d(v, v_2) + 1$
        $T \leftarrow T \cup \{(SLeaf(c), v_2)\} - \{(u_2, v_2)\}$
    else
        $t \leftarrow$ a node such that $d(r, t) = d(r, u) + d(v, v_2) + 1$ and $P_T(u, t) \subset P_T(u, SLeaf(c))$
        $T \leftarrow T \cup \{(t, v_2)\} - \{(u_2, v_2)\}$
    if $\{(u_2, x) | (u_2, x) \in T\} = \varnothing$
        $T \leftarrow T \cup \{(SLeaf(c), v)\} - \{(u, v)\}$
else
    $T \leftarrow T \cup \{(u_1, u_2), (v_1, v_2)\} - \{(u_1, v_1), (u_2, v_2)\}$
    $\forall (x, y) \in P_T(v_1, u_2)$, $T \leftarrow T \cup \{(y, x)\} - \{(x, y)\}$
```

Fig. 4. Removal of an intra-path shared bottleneck

In the case of an intra-path shared bottleneck, the shallowest leaf of v_1 may be v_2 itself or a node in its subtree. If v_1's shallowest leaf is not in v_2's subtree, REMOVE-INTER-PATH-SHARED-BOTTLENECK is applied to attach v_2 to the shallowest leaf of u_1. Otherwise, we have two cases in Line 4, depending on whether there is any branch between v_1 and v_2. If there is, v_2's subtree is attached under that branch similarly as in an inter-path shared bottleneck case. Otherwise, it becomes a child of a node in the middle so that the depth of v_2 in increased at most by one. As in Lemma 1, this ensures that the leaf distance vector before REMOVE-INTRA-PATH-SHARED-BOTTLENECK precedes that after REMOVE-INTRA-PATH-SHARED-BOTTLENECK.

If there is no branch between v_1 and v_2, edges (u_1, v_1) and (u_2, v_2) are replaced with (u_1, u_2) and (v_1, v_2), and the edges in the middle are reversed so that the flow traverses in the opposite direction in Lines 15–16. Shortest-path routing guarantees that this reduces the total cost.

From the two cases above, we conclude the following lemma. Its proof is in [6].

Lemma 2. *Let D and D' denote leaf distance vectors before and after* REMOVE-INTRA-PATH-SHARED-BOTTLENECK *respectively, and C and C' be total costs before and after* REMOVE-INTRA-PATH-SHARED-BOTTLENECK. *Then we have either $D \prec D'$ or $D = D'$ and $C < C'$.*

3.3 Shared Bottleneck Elimination

Using the previous two lemmas, we prove that our algorithm removes all shared bottlenecks from a multicast tree.

Theorem 1. *By applying* REMOVE-INTER-PATH-SHARED-BOTTLENECK *or* REMOVE-INTRA-PATH-SHARED-BOTTLENECK *iteratively, all shared bottlenecks will be removed in a finite number of iterations.*

Proof. If there exists a shared bottleneck in a multicast tree, either REMOVE-INTER-PATH- or REMOVE-INTRA-PATH-SHARED-BOTTLENECK can always be applied to remove it. Each of them changes the leaf distance vector or decreases the total cost while maintaining the same leaf distance vector by Lemma 1 and Lemma 2. For a leaf distance vector D, there are only a finite number of leaf distance vectors D' such that $D \prec D'$. And the total cost C can be reduced only a finite number of times because it is lower-bounded, and each time the amount of reduction is also lower-bounded by the minimum link cost, which is a non-zero constant. Therefore, all shared bottlenecks are removed within a finite number of iterations. □

Note that our algorithms remove shared bottlenecks, providing that the available bandwidth B and cost c of each link remain constant. In practice, however, since available bandwidth keeps varying and the set of participating hosts H changes, the multicast tree that the algorithm converges to may also change. Nevertheless, we believe that the algorithm that converges to the desired target in a static environment is a good starting point for a dynamic environment, and we will show empirically that the actual protocol based on our algorithm is in fact able to adapt as available bandwidth and node membership changes occur.

4 Protocol Implementation

Our algorithm is based on assumptions that a shared bottleneck is detectable, that information such as shallowest nodes and ramification points is available, and that each execution of Remove-Inter-Path-Shared-Bottleneck or Remove-Intra-Path-Shared-Bottleneck does not interfere with another execution. In this section, we explain how these assumptions can be satisfied in a protocol implementation. In addition, we briefly describe how our protocol handles node joins and leaves in a dynamic environment.

4.1 Shared Congestion Removal

In real networks, a shared bottleneck is a congested link shared by multiple flows belonging to the same multicast session. DCW [2] determines whether two flows are sharing such "shared congestion" with high accuracy ($> 95\%$) if one-way delay for each flow is measured for 10 seconds with a sampling frequency of 10 Hz. It tolerates a synchronization offset up to one second between different flows, which is achievable with loose synchronization among participating nodes as follows.

For loose synchronization, each non-root node sends a packet to its parent periodically, and the parent replies with a timestamp. On receiving the reply, the node calculates the round-trip time and sets its clock to the timestamp plus half the round-trip time.

Shared congestion is detected and removed on a round-basis. The start time of each round is publicized by the root node; each node obtains information on the epoch T_0 and round interval T_r from its parent, and starts a round at $T_0 + nT_r$ with its local clock, where n is an integer. In every round, a node performs the following three tasks sequentially. (i) At the beginning of each round, one-way delay from a node to each of its children that are experiencing congestion is sampled for 10 seconds with a frequency of 10 Hz as recommended for DCW [2]. (ii) After measurement, a node waits for reports from all child nodes; the reports contain delay samples of edges experiencing congestion in the subtree of the corresponding child node. Once all reports are received, the node selects edge pairs such that the edges in each pair share a bottleneck link with each other. The node must ensure that executions of the bottleneck removal algorithms do not interfere with each other. Since bottleneck elimination relocates nodes in the subtree of $Ram(v_2)$ only, the node can select as many pairs as it can, as long as such subtrees of selected pairs do not overlap. Then, among all congested edges in its subtree, the node reports delay samples of those edges that "would not interfere" with selected pairs. Because edges involved in removing a bottleneck are (u_1, v_1) and those in the subtrees of $Ram(v_2)$ and v_2 only, shared bottlenecks in other edges can be removed concurrently. Therefore, the node sends to its parent the delay samples of those congested edges that are not involved in removing a bottleneck of any selected pair. (iii) Finally, the node removes shared bottlenecks in its subtree by running the algorithm for every selected pair.

4.2 Information Update

The algorithm requires that each node v should know $d(r, v)$, $SLeaf(v)$, and $Ram(v)$. These values are updated at each node u by exchanging information with its parent and with its child nodes when the values change. An information update packet from a parent u and a child v contains $d(r, u)$ and $Ram(v)$, which are used to update v's local information on $d(r, v)$ and $Ram(v)$. Similarly, an information update packet from a child v to its parent u contains $SLeaf(v)$, and u updates $SLeaf(u)$, $d(u, SLeaf(u))$, $SLeaf(v)$, $d(u, SLeaf(v))$, and the child node whose subtree $SLeaf(v)$ belongs to.

4.3 Membership Management

We assume that a joining node obtains the address of the root node through an out-of-band channel, such as WWW. When it sends a join request to the root node, it is accepted as a temporary child. If the new node does not experience congestion during the next round, it becomes a permanent child. Otherwise it is forwarded to one of the existing children of the root node. This procedure is propagated along the tree until the joining node becomes a permanent child of an existing node. One concern is that congestion caused by a temporary child may affect other children. This can be avoided if a parent node uses a priority queue for its outgoing flows, in which packets to the temporary child have lower priority than others.

When a node leaves, its children become temporary children of the parent of the leaving node. Then the temporary children are treated as joining nodes. Node failures are handled in the same way.

5 Evaluation

To evaluate our protocol, we compare it against two heuristic-based schemes. The first one optimizes the multicast tree using bandwidth estimation as in Overcast [7]. Each node estimates available bandwidth from the grandparent, parent and its siblings using 10 kB TCP throughput, and then relocates below the one with the highest estimation. However, 10 kB TCP throughput does not have very strong correlation with available bandwidth. Taking into account that a path chosen using 10 kB TCP throughput provides only half of the bandwidth of the best path [8], we optimistically assume that the bandwidth estimation has maximum error of 20%.

The second heuristic scheme is based on delay measurement. It is similar to the bandwidth-based one except that it selects the node with the shortest delay instead of the highest bandwidth and that the number of children each node can have is limited to four to avoid high fan-out.

For fair comparison, we also introduce errors in shared congestion detection. Since our goal is to show that our protocol performs better than heuristic-based ones, we conservatively assume that the detection error is 5%, which is higher than actual error rate (almost zero when measurement interval is longer than

10 seconds) of DCW [2]. Then we measure performance of each scheme using a flow-level simulator we wrote, where bandwidth allocation to flows is max-min fair. The relationship between the tree performance and error rate will also be presented.

5.1 Tree Performance Comparison

To demonstrate tree performance under heavy load, we run simulations on a network with a dense receiver population. The network topology is generated with GT-ITM [9]. There are 24 transit routers, 576 stub routers, and 1152 hosts participating the multicast session. The bandwidth (Mbps) of each link is randomly drawn from four different intervals: $[300, 1400)$ between transit routers, $[40, 70)$ between a transit and a stub router, $[5, 15)$ between stub routers, and $[1, 5)$ between a stub router and an end host. The source rate is set to 1 Mbps. Initially, the tree consists of the source (root) host only. All the other hosts then join the tree.

We use the following metrics to evaluate a multicast tree.

Link Stress. The number of flows in a multicast session that traverse a physical link. Defined in [10]. Link load] The sum of required rates for all flows in a link divided by the bandwidth of the link.

Relative Delay Penalty (RDP). The ratio of the delay from the root to a node in a tree to the unicast delay between the same nodes. Defined in [10].

Receiving Rate. The max-min fair rate assigned to a flow from the root to a node divided by the maximum rate of the flow (the source rate).

Below we show distributions of these metrics for trees built with the three different schemes: delay heuristic, bandwidth heuristic, and our bottleneck-free tree protocol. We run the bottleneck-free tree protocol until there is no shared congestion. The delay heuristic scheme is run until the tree does not change any more. However, the bandwidth heuristic may oscillate as shown in Overcast [7] because changing tree topology affects bandwidth estimation. Since Overcast becomes relatively stable after 20 rounds, we run the bandwidth heuristic up to 30 rounds.

Figure 5 shows the link stress distribution for links used by the multicast session. Since the delay heuristic tends to build a tree well-matched with the underlying topology, its link stress is far better than other schemes. Note that the bottleneck-free tree shows the worst performance in terms of link stress. However, this does not necessarily mean that it is abusing the network, because having a large number of flows in a link (high link stress) is totally acceptable if the link has available bandwidth to accommodate all of them. The next figure shows this point clearly.

Figure 6 presents the distribution of link load, which is the amount of bandwidth required to carry all flows traversing a link divided by the link's available bandwidth. Contrary to the previous result, the delay heuristic is the worst

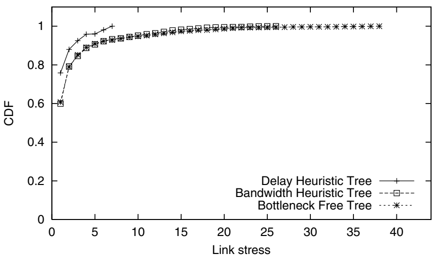


Fig. 5. Link stress distribution

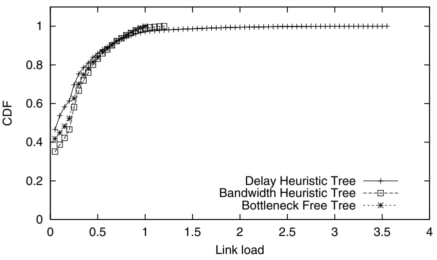


Fig. 6. Link load distribution

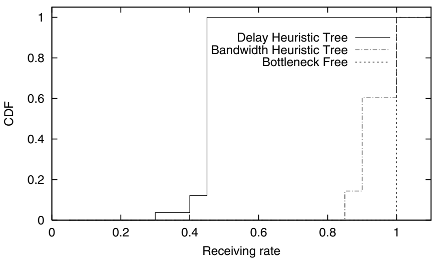


Fig. 7. Receiving rate distribution

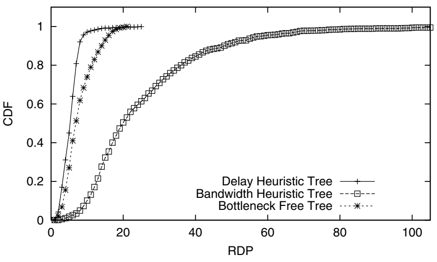


Fig. 8. RDP distribution

among the three; on some links, the required bandwidth to support the multicast session is more than 3.5 times the available bandwidth. This is because the delay heuristic often chooses a link with small bandwidth if the delay of a path going through the link is short. Note that link load with the bottleneck-free tree is always less than one. The bandwidth heuristic maintains similar performance as the bottleneck-free scheme, but some links have load higher than one. Since each of such links throttles receiving rates of the entire subtree connected upstream through the link, even a few of them may affect a large number of receiving nodes. In Fig. 7, the receiving rate distribution is shown. Due to high link load, all the receiving nodes in the delay heuristic tree receive less than half of the source rate. The distribution for the bandwidth heuristic shows the impact of the few links with high load in Fig. 6; only less than 40% of the receiving nodes can receive data at the full source rate. The other 60% experience quality degradation due to bandwidth shortage. However, in the bottleneck-free tree, 100% of the receiving nodes receive at the full source rate since it maintains link load less than one. Usually such gain in receiving rate comes with the cost of longer delay. However, our algorithm is very careful in changing the tree topology not to increase depth of a relocated node unnecessarily. As a result, its RDP is only a little worse than the tree built with the delay heuristic, as illustrated in the distribution of RDP in Fig. 8. Because the bandwidth heuristic pays little attention to delay, its RDP is worse than the others.

We have to mention that this experiment regarding RDP is somewhat unfair to the bandwidth heuristic; the relative delay of the bandwidth heuristic would be better if rate allocation was TCP fair rather than max-min fair. That is because TCP throughput is affected by round-trip time, and then by choosing a path with high throughput, a short path is very likely to be chosen. Nevertheless, since the 10 kB TCP throughput does not have strong correlation with round-trip time [8], we do not expect significant improvement with TCP fair rate allocation.

5.2 Convergence Speed

The next aspect of our protocol to evaluate is its convergence speed. The protocol defines a series of actions performed during each round, and thus we use *round* as a unit to measure the convergence time. The length of a round interval is on the order of tens of seconds, because delay measurements for shared congestion detection take ten seconds.

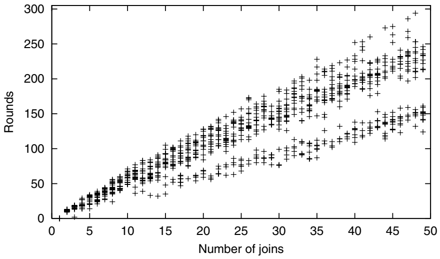


Fig. 9. Convergence after nodes join

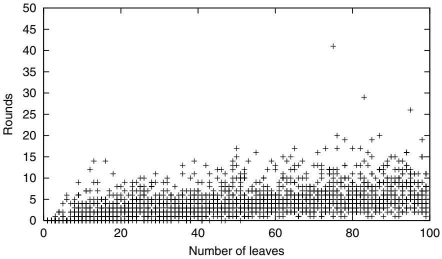


Fig. 10. Convergence after nodes leave

Fig. 9 shows how long it takes for a tree to stabilize when n nodes join. The convergence time increases linearly as the number of joining nodes increases. This presents an upper bound because in this scenario all the nodes first become children of the root node resulting in shared congestion on most links close to the root. The convergence time would be reduced if nodes are allowed to contact a non-root node directly to join or the root node forwards the new node to a random node, though the resulting tree might be taller.

Unlike joins, concurrent leaves can be handled relatively easily. In Fig. 10, we plot the convergence time when n hosts leave the tree. Except for a few outliers, most cases take less than 20 rounds. This is because shared bottlenecks in different subtrees can be eliminated concurrently.

We plot in Fig. 11 the time it takes to remove bottlenecks with different depth in the tree. A new shared bottleneck was created by reducing available bandwidth. As we can expect, a bottleneck near a leaf (depth larger than 25)

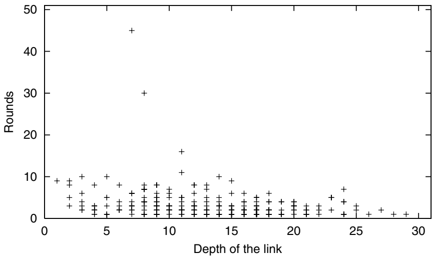


Fig. 11. Convergence after available bandwidth change

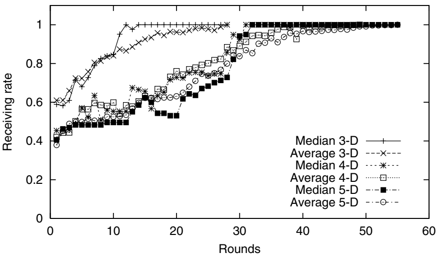


Fig. 12. Receiving rate increase as a tree converges

can be removed within a couple of rounds. On the other hand, a bottleneck close to the root takes longer—up to ten rounds with a few outliers.

Figure 12 demonstrates receiving rate changes as time elapses until the tree converges to the bottleneck-free state. The initial trees are built randomly with fixed degree. For each degree (3, 4, or 5), both median and average receiving rates are plotted. Although it takes tens of rounds to converge, most receiving rate increase is achieved within early half of the convergence time.

5.3 Effects of Measurement Errors

Since all the three schemes we evaluated in Sect. 5.1 depend on network measurements, we also investigated the relationship between errors in measurements and tree performance. For the bandwidth-heuristic tree, we changed the measurement error from 0% to 32%, but the difference was insignificant. For shared congestion detection, there are two types of errors: false positive and false negative. We changed these values from 0% to 6%, which is higher than the error rates observed in measurements [2], but could not find noticeable changes in terms of RDP and convergence speed. Detailed results are shown in [6].

6 Conclusion

In bandwidth-demanding multicast applications such as multimedia distribution, it is critical for a user to receive at the full source rate not to experience quality degradation. Though many heuristics to achieve high receiving rate have been proposed, they often fail to provide the required receiving rate for many users. We proposed a new tree construction algorithm that removes bottlenecks caused by the multicast session, and proved that it removes every such bottleneck. If the available bandwidth of each link is larger than the source rate, the algorithm guarantees that all receiving nodes receive at the full source rate. Simulation results show that our protocol maintains low link load and short delay penalty while providing the maximum receiving rate.

References

1. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scribe: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications **20** (2002) 100–110
2. Kim, M.S., Kim, T., Shin, Y., Lam, S.S., Powers, E.J.: A wavelet-based approach to detect shared congestion. In: Proceedings of ACM SIGCOMM 2004. (2004)
3. Harfoush, K., Bestavros, A., Byers, J.: Robust identification of shared losses using end-to-end unicast probe. In: Proceedings of the 8th IEEE International Conference on Network Protocols. (2000)
4. Katabi, D., Bazzi, I., Yang, X.: A passive approach for detecting shared bottlenecks. In: Proceedings of the 10th IEEE International Conference on Computer Communications and Networks. (2001)
5. Rubenstein, D., Kurose, J., Towsley, D.: Detecting shared congestion of flows via end-to-end measurement. Transactions on Networking **10** (2002) 381–395
6. Kim, M.S., Li, Y., Lam, S.S.: Eliminating bottlenecks in overlay multicast. Technical Report TR–04–47, Department of Computer Sciences, The University of Texas at Austin (2004)
7. Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, M.F., O'Toole, Jr., J.W.: Overcast: Reliable multicasting with an overlay network. In: Proceedings of 4th Symposium on Operating Systems Design and Implementation. (2000) 197–212
8. Ng, T.S.E., Chu, Y., Rao, S.G., Sripanidkulchai, K., Zhang, H.: Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In: Proceedings of IEEE INFOCOM 2003. (2003)
9. Calvert, K.L., Doar, M.B., Zegura, E.W.: Modeling Internet topology. IEEE Communications Magazine **35** (1997) 160–163
10. Chu, Y., Rao, S.G., Seshan, S., Zhang, H.: A case for end system multicast. IEEE Journal on Selected Areas in Communications **20** (2002)

TOMA: A Viable Solution for Large-Scale Multicast Service Support*

Li Lao[1], Jun-Hong Cui[2], and Mario Gerla[1]

[1] Computer Science Dept., University of California, Los Angeles, CA 90095
[2] Computer Science & Engineering Dept., University of Connecticut, CT 06029

Abstract. In this paper, we propose a Two-tier Overlay Multicast Architecture (TOMA) to provide scalable and efficient multicast support for various group communication applications. In TOMA, Multicast Service Overlay Network (MSON) is advocated as the backbone service domain, while end users in the access domains form a number of small clusters, in which an application-layer multicast protocol is used for the communication between the clustered end users. TOMA is able to provide efficient resource utilization with less control overhead, especially for large-scale applications. It also alleviates the state scalability problem and simplifies multicast tree construction and maintenance when there are large numbers of groups in the networks. Simulation studies are conducted and the results demonstrate the promising performance of TOMA.

1 Introduction

Over the years, tremendous efforts have been made to provide multicast support, ranging from IP multicast to recently proposed application-layer multicast. IP multicast utilizes a tree delivery structure which makes it fast, resource efficient and scale well to support very large groups. However, IP multicast is still far from being widely deployed due to many technical reasons as well as marketing reasons. The most critical reasons include: the lack of a scalable inter-domain routing protocol, the state scalability issue with a large number of groups, the lack of support in access control, the requirement of global deployment of multicast-capable IP routers and the lack of appropriate pricing models [10, 1]. These issues make Internet Service Providers (ISPs) reluctant to deploy and provide multicast service.

In recent years, researchers resort to application-layer multicast approach: multicast-related features are implemented at end hosts [12, 9, 16, 3, 14, 19, 17, 20, 6]. Data packets are transmitted between end hosts via unicast, and are replicated at end hosts. These systems do not require infrastructure support from intermediate nodes (such as routers), and thus can be easily deployed. However, application-layer multicast is generally not scalable to support large

* This material is based upon work supported by the National Science Foundation under Grant No. 0435515 and Grant No. 0435230.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 906–917, 2005.

multicast groups due to its relatively low bandwidth efficiency and heavy control overhead caused by tree maintenance at end hosts. In addition, from the point of view of an ISP, this approach is hard to have an effective service model to make profit: group membership and multicast trees are solely managed at end hosts, thus it is difficult for the ISP to have efficient member access control and to obtain the knowledge of the group bandwidth usage, and makes an appropriate pricing model impractical, if not impossible.

Then the question is: what is the viable or practical solution for large-scale multicast support? In a multicast service, multiple parties are involved: network service providers (or higher-tier ISPs), Internet Service Providers (or lower-tier ISPs, which we refer to as ISPs for short in this paper), and end users. However, which party really cares about multicast? End users do not care as long as they can get their service at a reasonable price. This is why many network games are implemented using unicast. Network service providers do not care as long as they can sell their connectivity service with a good price. Obviously, ISPs in the middle are the ones who really care about multicast: their goal is to use limited bandwidth purchased from network service providers to support as many users as possible, i.e., to make a biggest profit. Therefore, to develop a practical, comprehensive, and profitable multicast service model for ISPs is the critical path to multicast wide deployment.

Strongly motivated, in this paper, we propose a **T**wo-tier **O**verlay **M**ulticast **A**rchitecture (called **TOMA**) to provide scalable, efficient, and practical multicast support for various group communication applications. In this architecture, we advocate the notion of **M**ulticast **S**ervice **O**verlay **N**etwork (**MSON**) as the backbone service domain. MSON consists of service nodes or proxies which are strategically deployed by an MSON provider (ISP). The design of MSON relies on well-defined business relationships between the MSON provider, network service providers (i.e., the underlying network domains), and group coordinators (or initiators): the MSON provider dimensions its overlay network according to user requests (provided by long-term measurement), purchases bandwidth from the network service providers based on their service level agreements (SLAs), and sells its multicast services to group coordinators via service contracts. Outside MSON, end hosts (group members) subscribe to MSON by transparently connecting to some special proxies (called *member proxies*) advertised by the MSON provider. Instead of communicating with its member proxy using unicast, an end host could form a cluster with other end hosts close by. In the cluster, application-layer multicast is used for efficient data delivery between the limited number of end users. The end users participating in the groups only need to pay for their regular network connection service outside MSON.

To make TOMA a reality, we face many challenges: 1. **Efficient management of MSON**: An MSON is expected to accommodate a large number of multicast groups. How does an MSON provider efficiently establish and manage numerous multicast trees? 2. **Cluster formation outside MSON**: Multicast group members may disperse around the world. How should they select and subscribe to appropriate member proxies? And how are efficient clusters formed among end users? 3. **MSON dimensioning**: Given the TOMA architecture, how should the MSON provider dimension the overlay network? In other words, where should the overlay proxies be placed, which links are used to transmit data, and how much bandwidth should be reserved on each link? 4. **Pricing**: The lack of feasible pricing schemes is one of the main barriers that delay the deployment of IP multicast. Therefore, how to charge the users of MSON is an important factor to decide whether MSON is a practical solution.

In this paper, we focus our efforts on investigating the first two issues (Note that we have also developed some effective mechanisms for overlay dimensioning and pricing [15], which are not presented here due to space limit). To address the management of MSON, we propose an efficient and scalable protocol, called OLAMP (OverLay Aggregated Multicast Protocol), in which we adopt *aggregated multicast* approach [11], with multiple groups sharing one delivery tree. Outside MSON, we develop efficient member proxy selection mechanisms, and choose a core-based application-layer multicast routing approach for data transmission inside clusters. We conduct extensive simulation studies and show the promising performance of TOMA: it provides efficient multicast service comparable to IP multicast; it outperforms NICE, a representative application layer multicast protocol, in the simulated scenarios; and it is scalable to group size as well as to the number of co-existing groups.

The rest of this paper is organized as follows. We review some background and related work in Sect. 2. In Sect. 3, we present an overview of TOMA architecture and address the critical issues of MSON management and cluster formation outside MSON. In Sect. 4, we evaluate the performance of TOMA by extensive simulations. Finally, we summarize our contribution in Sect. 5.

2 Background

2.1 Aggregated Multicast

It is known that IP multicast is plagued from the state scalability issue. Multicast state scalability problem refers to the explosion of multicast state (i.e., memory to store multicast state in the routers) and control overhead (i.e., multicast tree setup and maintenance overhead when a "soft-state" approach is employed) for a large number of co-existing multicast groups. Aggregated multicast is proposed to improve multicast state scalability in transit (especially backbone) domain [11]. Observing that many multicast trees within a single domain are likely to have significant overlapping when there are numerous multicast groups, aggregated multicast forces multiple groups with similar shapes to share a single tree and reduces the number of multicast trees in the transit domain. Thus, the tree management overhead is significantly reduced, and the multicast state information stored in the routers is dramatically reduced too. In this paper, we design a protocol called OLAMP within MSON based on aggregated multicast.

2.2 Related Work

There is a large body of work on application-layer multicast [12, 9, 16, 3, 14, 19, 17, 20, 6]. In Yoid [12], multicast trees are constructed by each member individually selecting a parent. In End System Multicast [9], end hosts cooperatively construct a multicast delivery tree on top of a mesh. ALMI [16] uses a centralized entity to collect membership information and to periodically calculate a minimum spanning tree. TAG [14] exploits the underlying network topology to optimize multicast trees.

Recently, the notion of infrastructure-supported overlays has received increasing attentions. Example seminal works are Overcast [13] and RMX [8]. Both of them use overlay nodes to support multicast routing, and their main focus is on building reliable multicast trees. In the literature, several two-tier architectures have also been proposed for scalable multicast support (such as OMNI [4], MSN [18], and Scattercacst [7]). However, these architectures are based on different service models from ours. The foundation of TOMA is well-defined business relationships between MSON providers, network service providers, and group coordinators. Accordingly, we focus on a number of new challenges faced by MSON providers, such as scalable MSON management, efficient cluster formation and MSON dimensioning. Moreover, in [18, 4], the authors focus on single multicast tree optimization and endeavor to optimize end-to-end delay and access bandwidth usage at service nodes. In our work, we put our efforts on the scalability issues for multiple co-existing groups. To our best knowledge, this paper is the first work to address the multicast state scalability issue in overlay networks.

3 TOMA: A Two-Tier Overlay Multicast Architecture

3.1 TOMA Overview

In the TOMA architecture, MSON is advocated as the service backbone domain. In MSON, overlay proxies are strategically placed to form an overlay network, on top of which multicast distribution trees are built for data delivery. Since an MSON provider always aims to have a bigger customer base and make a bigger profit, the MSON management scheme should be scalable to the group size (i.e., the number of members) as well as the number of groups. In this paper, we propose a protocol called OLAMP for efficient and scalable MSON management. In OLAMP, we adopt aggregated multicast [11]. Data packets are encapsulated at incoming proxies, transmitted on aggregated trees, and decapsulated at outgoing proxies. Outside MSON, end users subscribe to MSON by transparently connecting to **member proxies** advertised by the MSON provider. Each member proxy organizes some users into a “cluster”, where an application-layer multicast tree (also denoted as peer-to-peer or P2P multicast tree in this paper) is formed for data delivery among the cluster members.

Each TOMA group is identified by a URL-like name *TOMA://groupname.xyzmson.com/*. End users (sources and receivers) explicitly send out a subscription request containing the group name, which will reach the DNS server and **group registry server** of the MSON. The group registry server enforces member access control policy and maintains group membership information. The DNS server will send back to the subscriber a list of IP addresses of the advertised member proxies, from which a member proxy will be selected.

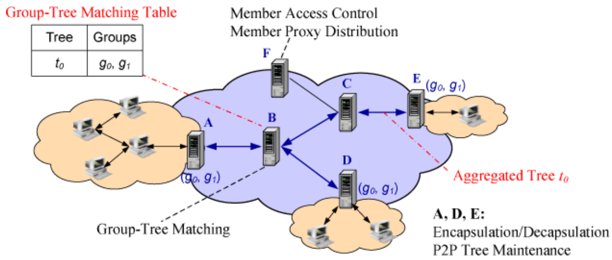


Fig. 1. A big picture of TOMA, where F is the group registry server/DNS server, B is the host proxy for groups g_0 and g_1, A, D and E are member proxies, and groups g_0 and g_1 share the aggregated tree t_0

After finding a member proxy, the end user sends its join message to the member proxy, which will then subscribe to the multicast group inside the MSON on behalf of this member. The member proxy will set up a P2P multicast tree in the local cluster and relay the join request to a predetermined proxy (referred to as **host proxy** for group g) to establish the overlay multicast tree in the backbone domain using OLAMP.

In the backbone domain, each group is managed by a host proxy. After receiving a join request for group g, this host proxy conducts multicast routing and group-tree matching (i.e., the procedure of assigning groups to aggregated trees) mechanisms to map group g to an aggregated tree. Considering that bi-directional trees can be used to cover groups irrespective of the location of sources, we use bi-directional aggregated trees to further reduce the number of aggregated trees. The host proxy for group g can be randomly selected or by applying a hash function (called *group-to-host* hashing function). Note that group member dynamics (i.e., join or leave of individual end users) generally do not affect the tree aggregation: only when a cluster joins a group for the first time or completely leaves a group, the member proxy needs to send join or leave request to the corresponding host proxy.

In a nutshell, in TOMA, member proxies manage P2P multicast trees in their clusters, host proxies conduct group-tree matching, and OLAMP connects member proxies and host proxies, efficiently managing aggregated multicast trees in the backbone domain. It is also possible to have some OLAMP-aware **“pure” forwarding proxies** whose main responsibility is to forward multicast data inside the backbone domain. A big picture of TOMA is illustrated in Fig. 1. In the following, we address some major design issues of TOMA in more details.

3.2 OLAMP for Efficient MSON Management

In MSON, the state scalability issue is even worse than IP multicast since the capability of proxies is usually far below that of IP routers. If a proxy manages large numbers of multicast trees, it has to maintain large forwarding tables and thus causes lookup speed to be slowed down. Furthermore, if a soft state approach is used, heavy tree management overhead due to multicast refresh messages will be incurred. Therefore, we design OLAMP to provide scalable and efficient MSON management.

OLAMP is manipulated among member proxies, host proxies, as well as "pure" forwarding proxies. To facilitate our description, we denote the control messages in OLAMP as O-type messages, which include *O-JOIN*, *O-JOIN-ACK*, *O-LEAVE*, *O-LEAVE-ACK*, *O-SWITCH*, *O-GRAFT* and *O-PRUNE*. Essentially, in MSON, every proxy is capable of handling O-type messages and maintains a multicast routing table.

Member (proxy) Join and Leave. When a member proxy *mp* decides to relay a join request for group g, it sends *O-JOIN(g)* to the host proxy *hp* of g. After conducting the group-to-tree matching, the host proxy *hp* finds or computes an appropriate aggregated tree, say, t. It will send back an *O-JOIN-ACK(g, t)* message to *mp*. If *mp* has not joined the delivery tree t, it will graft to the tree by sending an *O-GRAFT(t)* message towards *hp*, and the proxies along this propagation path will update their routing tables accordingly.

Similarly, when a member proxy *mp* discovers that no end users are connected, *mp* sends an *O-LEAVE(g)* message to the host proxy *hp*, which may trigger a group-tree matching process. If no other member proxies belong to group g, the host proxy will remove the group-tree mapping between g and t. This group-tree matching may trigger removal of the tree t when there are no other groups mapped onto t. In this case, *hp* sends an *O-LEAVE-ACK(t)* messages to the tree leaves of t, which will in turn prune from the tree by sending *O-PRUNE(t)* towards *hp*.

A Dynamic Group-Tree Matching Algorithm. When an aggregated tree is bigger than a group, data will be delivered to non-member nodes, leading to bandwidth waste. Obviously, there is a trade-off between bandwidth waste and aggregation: the more bandwidth we are willing to sacrifice, the more groups can share one tree, and thus the better aggregation we can achieve. Hence, it is necessary to control the amount of bandwidth waste in group-tree matching. Assume that an aggregated tree t is shared by groups $g_i, 1 \leq i \leq n$, each of which has a native tree $t_0(g_i)$ (a native tree of a group is a "perfect" match for that group and it can be computed using multicast routing algorithms). Then the **average percentage bandwidth overhead** for t can be defined as

$$\delta(t) = \frac{\sum_{i=1}^{n} B(g_i) \times (C(t) - C(t_0(g_i)))}{\sum_{i=1}^{n} B(g_i) \times C(t_0(g_i))} = \frac{C(t) \times \sum_{i=1}^{n} B(g_i)}{\sum_{i=1}^{n} B(g_i) \times C(t_0(g_i))} - 1, \quad (1)$$

where $C(t)$ is the cost of tree t (i.e., the total cost of the links on tree t), and $B(g)$ is the bandwidth requirement of group g.

When a host proxy *hp* receives an *O-JOIN(g)* message, it updates the corresponding entries of multicast group table and executes the QoS-Aware group-tree matching algorithm (Algorithm 1), in which we denote the given bandwidth overhead threshold as b_{th}. It works as follows: if g is not new and the current tree t for group g is still appropriate (i.e., t can cover all members of g with enough bandwidth and the bandwidth overhead is within the threshold b_{th}), t

Algorithm 1 GTMatch(g, t, T_e, b_{th})

```
//$t$ is current tree for $g$, $T_e$ is the set of all existing trees
if $t$ is not null AND $t$ covers $g$ AND $\delta(t) \leq b_{th}$ then
  return $t$
else
  $T_c \leftarrow \emptyset$ //$T_c$ is the set of candidate trees
  compute a native multicast tree $t_0$ for $g$
  for $t \in T_e$ do
    if $t$ covers $g$ AND $\delta(t) \leq b_{th}$ then
      $T_c \leftarrow T_c \cup \{t\}$
    end if
  end for
  if $T_c$ is not empty then
    return $t \in T_c$ with min $C(t)$
  else
    return $t_0$
  end if
end if
```

is used for g. In all other cases, check if any existing tree is appropriate for g. If so, the one with the minimum cost is selected. Otherwise, the native tree t_0 is used to cover g (if t_0 does not exist due to bandwidth constraint, g has to be rejected). After mapping g to a tree t', *hp* sends *O-JOIN-ACK(g, t')* back to *mp*. If g is not a new group and $t \neq t'$, an *O-SWITCH(g, t, t')* message is sent via multicast (using tree t) to all other members of g to switch g from tree t to t'.

3.3 Cluster Formation Outside MSON

Member Proxy Selection. On receiving a list of candidate member proxies from the MSON DNS server, an end user selects one proxy based on the criteria of low latency and low workload (in terms of processing power and bandwidth availability), since low latency is critical to real-time applications and lightly-loaded proxies tend to have more resources. The end user measures the RTT (round-trip time) to candidate proxies by sending *ping* requests. In the reply message to a *ping* request, the proxy piggybacks its workload information, e.g., the total number of end users it currently handles or the total amount of access bandwidth in use. If there are multiple proxies close by, the one with the lowest workload is selected by the user.

P2P Multicast in Access Networks. Outside MSON, the end users associated with the same member proxy form a cluster. In a cluster, nodes communicate in a P2P fashion via application-layer multicast trees: when an end user sends packets to the group, the packets are transmitted to all other end users in the same cluster and to the member proxy as well. The member proxy will relay these packets to other member proxies (at the edge of the MSON), which will in turn deliver the data to the group members in their clusters.

Due to the existence of a member proxy node in every cluster, we adopt a core-based approach (similar to ALMI [16]) to construct P2P multicast trees. In the cluster, the member proxy acts as a core, storing the group membership information in its cluster scope. Periodically, end users monitor its peers in the same cluster regarding the path quality (such as delay and available bandwidth), and report this information to its member proxy. After putting together the global picture of its clusters, the member proxy computes P2P multicast delivery trees and disseminates the $(parent, children)$ entries to the members. Finally, end users connect with their children and transmit data packets via unicast. If a member leaves ungracefully, its peers will detect this from periodic probing and the multicast tree will be repaired by the member proxy.

4 Performance Evaluation

In this section, we conduct simulation experiments in NS-2 to evaluate the performance of TOMA. We compare TOMA with a scalable application-layer multicast protocol NICE [3], an IP multicast protocol (Core-Based Tree [2]), and unicast protocol in various network topologies. NICE [3] scales to large groups in terms of control overhead and logical hops by recursively arranging group members into clusters to form a hierarchical overlay topology, which implicitly defines a source-specific tree for data delivery. In comparison with NICE, we find that TOMA can achieve very competitive performance for large groups.

4.1 Simulation Settings

To comprehensively evaluate TOMA, we use two types of network topologies and group membership models. The first type is synthetic Transit-Stub (TS) topologies [5] with 50 transit domain routers and 500−2000 stub domain routers. End hosts are attached to stub routers uniformly at random. The second type of topology is abstracted from a real network topology, AT&T backbone, and it consists of 54 routers. To each router i, we assign a weight w_i and randomly attach end hosts to the router with a probability proportional to w_i [15].

For overlay dimensioning, we use some simple heuristics to determine proxy locations and overlay links. We randomly select 80% of the transit nodes (i.e.,

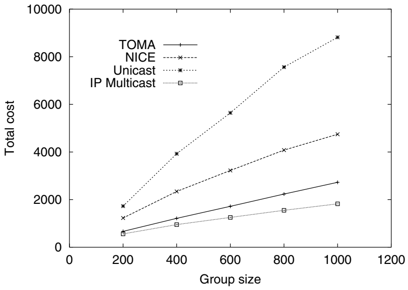


Fig. 2. Tree cost vs. group size

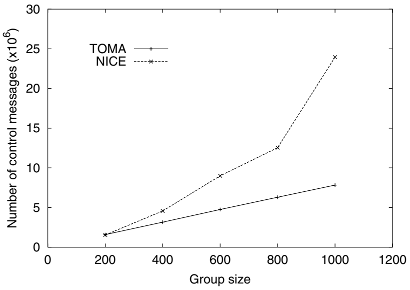


Fig. 3. Control Overhead vs. group size

40 nodes) in TS topologies and 8 gateway routers in AT&T topologies as proxy nodes. Note that proxy nodes can be co-located with either transit (i.e., gateway) or stub (i.e., access) nodes. We choose proxy nodes from transit nodes and gateway routers because they usually have high degree and are located in the core of the network. Between every pair of proxies, an overlay link is established if the shortest IP-layer path between them does not go through any other proxies. As a result, the constructed overlay networks tend to resemble the underlying network topologies and thus have high routing efficiency. It is worth noting that our simulation results in [15] demonstrate that overlay networks dimensioned with more systematic approaches can achieve even better performance.

4.2 Multicast Tree Performance

In simulation experiments, we focus on large group sizes of 200 to 1000 members to test the scalability of TOMA. End hosts join the multicast group during an interval of 400 seconds, and the session ends at 1000 seconds. We collect the metrics after the multicast tree has stabilized. Due to space limitation, we only present the results for TS topologies since AT&T yields similar results [15].

In Fig. 2, we plot the average tree cost (defined as the total number of links in a multicast tree) of these protocols as group size increases. As a reference, we also include the total link cost for unicast. Clearly, the performance of TOMA is comparable to IP multicast. In addition, TOMA outperforms NICE in all cases, and their difference magnifies as group size is increased. This efficiency gain of TOMA vs. NICE is due to two reasons. First, TOMA takes advantage of the carefully dimensioned overlay network which resembles the underlying network topology, whereas NICE relies on path probing techniques and constructs overlay topologies with degraded quality. Second, by using proxies as intermediate nodes, TOMA allows the packets to be replicated at proxies and hence decreases the redundant packets sent over the physical links. We also found out that TOMA achieves higher performance than NICE in several other metrics (e.g., link stress and path length), which is again due to the efficient overlay construction [15].

4.3 Control Overhead

In this subsection, we compare the total number of control messages generated by TOMA and NICE during a group's life time, i.e., 1000 seconds. For the sake of a fair comparison, we set the parameters in NICE and TOMA to the same values whenever possible. We found out that in NICE, the total number of control messages increases very rapidly with group size, while in TOMA, the increase is much more steady (Fig. 3). At group size of 1000, TOMA generates only about one third as many control messages as NICE. The reason is simple: in TOMA, the local clusters remain rather static and a member stays in the same cluster in spite of the behaviors of other members. However, in NICE, as members join and leave, the clusters split, merge, or change cluster leaders very frequently to enforce the bounds on cluster size, inducing numerous control messages. Furthermore, a NICE node needs to periodically send "heartbeat" messages to all other cluster members (or even to multiple clusters in the case

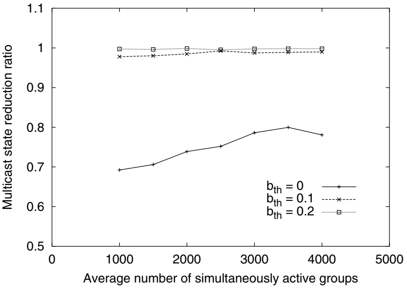


Fig. 4. State reduction

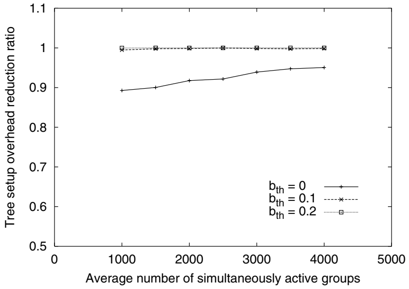


Fig. 5. Tree setup overhead reduction

of cluster leaders) to maintain the clusters, whereas a TOMA node only needs to refresh with its parent to maintain the connection.

4.4 Effectiveness of OLAMP

Recall that OLAMP not only reduces multicast forwarding entries, but also suppresses a lot of control messages transmitted for tree setup and maintenance. Therefore, we first measure *reducible multicast forwarding entries*, that is, the multicast state entries in non-member proxies, since member proxies always need to maintain group state for encapsulation and decapsulation purpose. We then measure *multicast tree setup overhead*, defined as the total number of tree setup messages relayed by proxies when multicast trees are established inside MSON.

We perform simulations for multiple groups in AT&T topology. We assume multicast groups arrive as a Poisson process, and their lifetime has an exponential distribution. We vary the bandwidth overhead threshold b_{th} in the group-tree matching algorithm presented in Sect. 3.2 from 0 to 0.2 in the simulation.

We define reduction ratio of multicast state entries as the percentage of reducible multicast state entries reduced by OLAMP. Then we plot this metric vs. number of active groups in Fig. 4. Clearly, more than 70% of the reducible multicast state entries can be reduced even when b_{th} is 0. We also observe that as more multicast groups become active, the reduction ratio improves. Furthermore, when b_{th} is raised from 0 to 0.2, we are able to eliminate more than 99% of the reducible state, albeit at the expense of higher bandwidth overhead. This figure demonstrates that TOMA is scalable to the number of co-existing groups.

We plot the reduction ratio of multicast tree setup overhead in Fig. 5. This figure exhibits a similar trend as Fig. 4: more reduction in tree setup overhead is accomplished when there are more concurrent groups and more bandwidth is wasted to force higher aggregation. We obtained similar results for tree maintenance overhead [15]. These results indicate that the MSON ISP can control the trade-off of bandwidth cost versus control overhead by setting b_{th} properly.

4.5 Summary

Our simulation results can be summarized as follows: TOMA creates multicast trees with performance almost comparable to IP multicast; the control overhead of TOMA is significantly less than NICE for large groups; TOMA is scalable to large numbers of groups in terms of control overhead and multicast state.

If we recall the architecture difference between TOMA and NICE, it is not difficult to draw the above conclusions. First of all, the strategic MSON dimensioning tends to take physical network topologies into consideration, and thus provides efficient multicast data delivery. In addition, the control overhead of TOMA is greatly reduced, since the communication is limited inside each local cluster and inside the overlay network. Finally, aggregated multicast approach further reduces the tree maintenance overhead and makes TOMA scalable to large numbers of groups. In contrast, in application layer multicast, the control overhead of exchanging information between peers and establishing multicast trees will increase rapidly with group size. Due to limited bandwidth at end systems, as group size increases, the depth of the multicast trees has to be increased, leading to long latency for data delivery. Nevertheless, we want to point out that TOMA requires infrastructure support to achieve the above benefits, which is a trade-off of TOMA vs. NICE.

5 Conclusions

In this paper, we propose and develop a two-tier overlay multicast architecture (TOMA) to support group communication applications in an efficient and scalable way. Our contributions can be summarized as follows: (1) We advocate the notion of infrastructure-supported overlays to facilitate the deployment of multicast service. (2) We provide a viable architecture design of TOMA, which adopts MSON as the backbone service domain and P2P multicast in the access domains to achieve efficient resource utilization with reduced control overhead. (3) We develop OLAMP for MSON management. The control overhead for establishing and maintaining multicast trees are significantly reduced, and far less forwarding state needs to be maintained at proxy nodes. (4) By extensive simulation studies, we show that the performance of TOMA is very promising. We believe that the invention of our practical, comprehensive, and scalable multicast service model would significantly facilitate the multicast wide deployment.

Future Work. We plan to continue our work in two directions. First, more research efforts are needed to investigate heuristic and distributed algorithms for overlay dimensioning. Second, we would like to conduct more comprehensive performance evaluation of TOMA, by comparing with other application-layer multicast protocols, and applying other multicast routing algorithms (eg., those reported in [18] and [4]) to OLAMP.

References

1. K. Almeroth. The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. *IEEE Network*, Jan./Feb. 2000.
2. A. Ballardie. Core Based Trees (CBT version 2) multicast routing: protocol specification. *IETF RFC 2189*, Sept. 1997.

3. S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable application layer multicast. In *Proceedings of ACM SIGCOMM*, Aug. 2002.
4. S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of IEEE INFOCOM*, Apr. 2003.
5. K. Calvert, E. Zegura, and S. Bhattacharjee. How to model and internetwork. In *Proceedings of IEEE INFOCOM*, Mar. 1996.
6. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489 – 1499, Oct. 2002.
7. Y. Chawathe, S. McCanne, and E. A. Brewer. *An Architecture for Internet Content Distributions as an Infrastructure Service*, 2000. Unpublished, http://www.cs.berkeley.edu/yatin/papers/.
8. Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable multicast for heterogeneous networks. In *Proceedings of IEEE INFOCOM*, Mar. 2000.
9. Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.
10. C. Diot, B. Levine, J. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, Jan. 2000.
11. A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast: an approach to reduce multicast state. *Proceedings of Sixth Global Internet Symposium(GI2001)*, Nov. 2001.
12. P. Francis. *Yoid: Extending the Multicast Internet Architecture.* White papar, http://www.aciri.org/yoid/.
13. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*, Oct. 2000.
14. M. Kwon and S. Fahmy. Topology aware overlay networks for group communication. In *Proceedings of NOSSDAV'02*, May 2002.
15. L. Lao, J.-H. Cui, and M. Gerla. A scalable overlay multicast architecture for large-scale applications. Technical report, UCLA CSD TR040008. http://www.cs.ucla.edu/NRL/hpi/papers.html, Feb. 2004.
16. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.
17. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proceedings of NGC*, Nov. 2001.
18. S. Shi and J. S. Turner. Routing in overlay multicast networks. In *Proceedings of IEEE INFOCOM*, June 2002.
19. A. Sobeih, W. Yurcik, and J. C. Hou. Vring: a case for building application-layer multicast rings (rather than trees). In *Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, Oct. 2004.
20. S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV'01*, June 2001.

The Burstiness Behavior of Regulated Flows in Networks

Yu Ying[1], Ravi Mazumdar[2], Catherine Rosenberg[2], and Fabrice Guillemin[3,⋆]

[1] Dept. of ECE, Purdue University, West Lafayette, IN, 47906, U.S.A
yingy@ecn.purdue.edu
[2] Dept. of ECE, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada
{mazum, cath}@ece.uwaterloo.ca
[3] France Telecom R&D DAC/CPN, 2, Ave. Pierre Marzin, 22300, Lannion, France
{fabrice.guillemin@francetelecom.com}

Abstract. In this paper we study the impact of statistical multiplexing on leaky-bucket regulated traffic streams as they pass through the network. In particular we show that the burstiness of a flow is randomized as it transits through the nodes with mean equal to its initial burstiness value at the ingress. We then show that the random burstiness for a single flow converges to a constant equal to the initial value at the ingress when the flow is multiplexed with a large number of sources. The results do not depend on independence or homogeniety between flows. We conclude by providing some simulation results that confirm the theory.

1 Introduction

Within the past decade, there has been a lot of work on characterizing the behavior of traffic in large networks (see [1, 2, 3, 4, 5, 16] and the references therein). In particular, it is well known that Internet traffic is bursty although there is not a rigid definition for burstiness. To manage the traffic in networks, a convenient approach is to shape traffic offered by users according to the parameters of an envelope at the network access point. One of the simplest regulation mechanisms is the so-called leaky bucket mechanism, which has been studied in great depth in the context of providing guaranteed quality of service. Traffic offered by a user is in general regulated by a dual leaky bucket, one leaky bucket enforcing the peak rate π and another the mean rate ρ by using a bucket size σ. This regulator is referred to as a (σ, ρ, π) regulator and a formalism to study worst-case delay bounds called *network calculus* has been developed for leaky bucket regulated inputs. Network calculus (see for instance [4, 5, 6, 7, 8, 9]) provides a framework to compute end-to-end delays based on algebraic properties of the (min,+) algebra.

Network calculus is essentially a deterministic worst-case approach. This analysis yields conservative results for network resources provisioning because it ne-

[⋆] This work has been supported by NSF grants ANI #0073359 and #0099137.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 918–929, 2005.

glects the statistical properties of traffic flows as well as their possible statistical independence. Moreover, the statistics of flows are altered as they interact with other flows through the scheduling and multiplexing mechanisms in routers. Thus, even though we might precisely regulate flows at the ingress, the internal behavior is more difficult to characterize. In this paper, from the viewpoint of computing statistical performance metrics, such as the mean delay through a network and overflow probability at an internal node, we aim to characterize the statistical properties of regulated flows as they progress through the network. In particular, this paper focuses on how statistical multiplexing alters the behavior of single regulated flows. We investigate how the parameters used at the network access point are modified when the single stream traverses several multiplexing nodes, such that performance characteristics at each node can be computed. This leads us to characterize the traffic parameters of flows at the output of a multiplexing stage. Insights from the single flow case will also help subsequent study on the behavior of aggregated flow when a portion of the output flow from a queue goes to the same node.

Recently, there has been a lot of interest in analyzing the statistical properties of multiplexed regulated streams. In [16] it is shown that when a large number of homogeneous sources are multiplexed, a single output flow of a switch essentially keeps its initial statistical characteristics from a large deviations point of view. The authors in [17] provide a network decomposition framework based on the same idea. In [10], the authors consider many independently multiplexed regulated streams as one flow and obtain an effective burst parameter for the aggregated input in terms of a given overflow probability. In [7, 13] the authors obtain bounds for the tail distribution of the buffer occupancy but do not directly address the burstiness characterization; and the analysis is restricted to one queue. Authors in [14] and [15] define a notion of exponentially bounded burstiness (EBB) which yields bounds on the tail distributions of the buffer occupancies. This EBB notion is mainly studied for multiplexing a few flows. In [11, 12], the authors obtained tight bounds on the mean delay and overflow probabilities for multiplexed regulated streams using an extremal traffic characterization but in single queues. None of these earlier results provide insights on how the statistical nature of the burstiness of a data flow behaves as it passes through *multiple* nodes and is multiplexed with *many* other flows in each node.

In fact, when a flow passes through several multiplexing nodes, the burstiness parameter is altered at each node and can no more be described by a constant parameter. This leads us to introduce the concept of a stochastic burst size which manages to characterize the burstiness of each sample path of the traffic flow. Using the idea of stochastic burstiness, we study the alteration of the traffic characteristics of a single flow as it passes through several multiplexing nodes.

In particular, we derive the moments of the burstiness parameter and provide some information on its distribution. We show the mean of the burstiness parameter for a single source remains a constant equal to the initial value of the burst size. The distribution spreads as the number of multiplexing nodes increases, but this distribution converges to its mean if the contribution of this flow is negligibly small at each node. *Our results hold for heterogeneous network and does not require independence between flows.*

The efficacy of this main result is then studied by simulating the mean delay and overflow probability in a tandem queue scenario. Theoretical results are validated via simulations. It turns out that when the number of sources in the network is large and the contribution of individual sources is small, then taking the burst size as unchanged is a reasonable approximation as in the large deviations setting considered in [16]. Therefore it can be used to facilitate the designs of shapers as well as admission control schemes for IntServ/DiffServ networks.

The outline of the paper is as follows: Section 2 gives a general definition of the stochastic burstiness. Section 3 studies in detail the burst size model $\widetilde{\sigma}(s,t)$ for single flows. Statistical properties of this model are given. In section 4, we show the effectiveness of our burstiness model via numerical results. Finally, section 5 concludes the paper.

2 Problem Formulation

In this paper, all the traffic processes are defined on some filtered probability space $(\Omega, \mathcal{F}, \mathbb{P}, \{\mathcal{F}_t\})$ and are leaky-bucket regulated at the access point of the network. We say an input flow $I(t)$ is leaky-bucket regulated, if $\forall s \leq t \in \mathcal{R}^+$,

$$I(s,t) \leq \min\{\pi(t-s), \sigma + \rho(t-s)\}, \tag{1}$$

where π stands for peak rate, ρ for mean rate and burst size is denoted by σ.

Equation (1) actually defines a worst case envelope for a regulated flow. If we let $I(w,(s,t))$ denote the amount of data produced by one sample path $\omega \in \Omega$ of flow $I(t)$ in the time period of $[s,t]$, (1) then indicates that for all the sample paths in Ω and for any time period $[s,t]$, $I(\omega,(s,t))$ is confined by a deterministic linear sub-additive function. Therefore, when the burstiness of a traffic flow is measured by the burst size σ, this definition implies that the burstiness of this flow is the maximum value achieved over all possible sample paths of flow $I(t)$ and all time interval $[s,t]$, i.e.,

$$\sigma = \max_{s \leq t,\ \omega \in \Omega} \{I(\omega,(s,t)) - \rho(t-s)\}. \tag{2}$$

We believe this is the fundamental reason why the performance bounds given by the deterministic network calculus approach are conservative for dimensioning the network with statistical performance metrics. Furthermore, the stochastic properties of a flow may change after multiplexing inside the network; while this deterministic approach fails to characterize the multiplexing effect.

To emphasize the stochastic properties of $I(t)$, in particular, of its burstiness inside the network, we relax the above worst case condition and consider each sample path of $I(t)$ in each time interval $[s,t]$. We define our stochastic burstiness of a regulated flow by

$$\sigma_r(\omega,(s,t)) = (I(\omega,(s,t)) - \rho(t-s), 0)^+. \tag{3}$$

This original stochastic burst model keeps track of each sample path of traffic flow $I(t)$; hence it dynamically inherits the stochastic information of $I(t)$ as the flow passes through a network of queues. Meanwhile, it satisfies the fundamental burstiness property that for each sample path, σ_r confines the maximum time period during which a regulated flow can produce data at its peak rate ([4]). It is easy to see that the peak rate of an output flow is bounded by the capacity of the node and the average rate of the flow is unchanged in a stable network. Hence to compute the performance, it is necessary to characterize the probabilistic properties of this stochastic process $\sigma_r(s,t)$.

In this paper, we study the burstiness characterization of a single output flow $\widetilde{I}_i(t)$ which is the ith output stream from a previous node with traffic descriptor denoted by $(\widetilde{\sigma}_i, \widetilde{\rho}_i, \widetilde{\pi}_i)$. We aim to understand how the burstiness parameter $\widetilde{\sigma}_i$ is changed from that of the ith input stream after statistical multiplexing in the queues of previous nodes, such that the single output traffic flow can be stochastically regulated as

$$\widetilde{I}(s,t) \leq \min\{\widetilde{\pi}(t-s), \widetilde{\sigma}(s,t) + \widetilde{\rho}(t-s)\}.$$

We focus on the burst level phenomena that occurs in a time scale typical of an on-off source activity period rather than the inter-arrival periods of packets, we thus ignore the discrete nature of the packets arrivals and regard the input flows as continuous stationary fluid processes. We restrict our analysis to a stable network in which all the nodes have infinite buffer with work conserving FIFO scheduling and each flow is assigned a route without loop. Traffic flows are called "fresh" if they are regulated with fixed parameters $(\sigma_j, \rho_j, \pi_j)$. Such flows are usually inputs to the edge nodes of the network. Their burst sizes are known at the edge node via subscription or a signaling procedure. Note that independence between flows are not assumed in our analysis.

3 Burstiness of a Single Flow

In this section, we tag one specific flow, $I^i(t)$, at the entrance and characterize its burstiness behavior after it passes through several internal nodes.

3.1 Single Output Flow with Fresh Inputs

We will begin our discussion with the simplest case as shown in Fig. 1. In this scenario, N_1 number of fresh flows $\{I_1^j(t)\}_{j=1}^{N_1}$ enters Q_1 where flow j has pa-

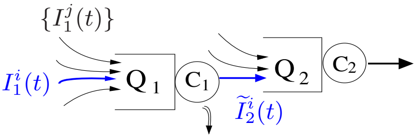


Fig. 1. Scenario S-I

rameters $(\sigma_1^j, \rho_1^j, \pi_1^j)$. We are interested in the burstiness process $\widetilde{\sigma}_2^i(s,t)$ of the ith output flow $\widetilde{I}_2^i(t)$.

To relate the output flow $\widetilde{I}_2^i$ with its input $I_1^i(t)$, we use the definition of Virtual Arrival Time Process (VATP) $v_1^i(t)$ ([9, 20]) of fresh input $I_1^i(t)$ as the following:

$$v_1^i(t) = \sup_{s \le t}\{s : I_1^i(s) \le \widetilde{I}_2^i(t)\}. \tag{4}$$

Hence $v_1^i(t)$ denotes the arrival time of a bit from source i to the queue which departs at time t. Since we deal with FIFO queueing, $v_1^i(t)$ can equivalently be defined as:

$$v_1^i(t) = \sup_{s \le t}\left\{s : s + \frac{Q_1(s)}{C_1} \le t\right\}. \tag{5}$$

where $Q_1(s)$ is the stationary workload of the first queue.

It readily follows that $v_1^i(t)$ is a nondecreasing right-continuous process and that $\widetilde{I}_2^i(s,t) = I_1^i(v_1^i(s), v_1^i(t))$. Then with $v_1^i(t) = t - \frac{Q_1(v_1^i(t))}{C_1}$, and the fact that $I_1^i(s,t)$ is regulated as

$$I_1^i(s,t) \le \min\{\pi_1^i(t-s), \sigma_1^i + \rho_1^i(t-s)\},$$

we obtain

$$\begin{aligned}\widetilde{I}_2^i(s,t) &\le \min\{C_1(t-s), \sigma_1^i + \rho_1^i(v_1^i(t) - v_1^i(s))\} \\ &= \min\{C_1(t-s), \rho_1^i(t-s) + \sigma_1^i + \frac{\rho_1^i}{C_1}[Q_1(v_1^i(s)) - Q_1(v_1^i(t))]\}.\end{aligned}$$

Here, we used the fact that the peak rate for the output flow is confined by the capacity C_1 of the first queue, $\widetilde{\pi}_2^i = C_1$. Hence $\widetilde{I}_2^i(s,t)$ is also regulated, and its burstiness parameter can be characterized as a stochastic process $\{\widetilde{\sigma}_2^i(s,t)\}$ defined by

$$\widetilde{\sigma}_2^i(s,t) = \sigma_1^i + \frac{\rho_1^i}{C_1}[Q_1(v_1^i(s)) - Q_1(v_1^i(t))]. \tag{6}$$

As seen from (6), the stochastic properties of $\widetilde{\sigma}_2^i(s,t)$ are determined by another random process $Q_1(v_1^i(t))$ which is a random-time shifting of stationary workload process $Q_1(t)$ of the first queue. The following lemma shows that, with FIFO queueing, this process $Q_1(v_1^i(t))$ is also a stationary process and it has the same moment distributions as $Q_1(t)$.

Lemma 1. *Consider several input streams $\{I^j(s,t)\}$ entering a FIFO queue with capacity C, with $\rho^j = \mathbb{E}[I^j(0,1)]$ and $\sum \rho^j < C$. Let $Q^*(t) = Q(v^i(t))$, where $v^i(t)$ is the Virtual Arrival Time Process. Then $Q^*(t)$ is a stationary process and for $n \ge 1$, $\mathbb{E}[(Q^*)^n(t)] = \mathbb{E}[Q^n]$.*

Proof. The proof follows mutatis mutandis from [20–Theorem 4].

Therefore, we can obtain the following statistical properties of $\sigma_2^i(s,t)$:

Proposition 1 (Properties of $\widetilde{\sigma}_2^i(s,t)$). *For any sample path, $\widetilde{I}_2^i(s,t)$ is regulated with $(\widetilde{\sigma}_2^i(s,t), \rho_1, C_1)$:*

$$\widetilde{I}_2^i(s,t) \leq \min\{C_1(t-s), \rho_1^i(t-s) + \widetilde{\sigma}_2^i(s,t)\}. \tag{7}$$

The mean value of $\widetilde{\sigma}_2^i(s,t)$ is given by

$$\mathbb{E}[\widetilde{\sigma}_2^i(s,t)] = \sigma_1^i, \tag{8}$$

and its second moment is upper bounded.

Finally, we have $\widetilde{\sigma}_2^i(s,t) \to \sigma_1^i$ in probability as $\frac{\rho_i}{C_1} \to 0$.

Proof. Equation (7) is obtained from equation (6). From the same equation and Lemma 1,

$$\mathbb{E}[\widetilde{\sigma}_2^i(s,t)] = \mathbb{E}[\sigma_1^i + \frac{\rho_1^i}{C_1}[Q_1^*(s) - Q_1^*(t)]] = \sigma_1^i,$$

and equation (8) follows. Also, its second moment is bounded since

$$\begin{aligned}\mathbb{E}[(\widetilde{\sigma}_2^i(s,t))^2] &= (\sigma_1^i)^2 + 2\frac{(\rho_1^i)^2}{C_1^2}\mathbb{E}[Q_1^2] - 2\frac{\rho_1^i}{C_1}\mathbb{E}[Q_1^*(t)Q_1^*(s)] \\ &\leq (\sigma_1^i)^2 + 2\frac{(\rho_1^i)^2}{C_1^2}\mathbb{E}[Q_1^2].\end{aligned}$$

Finally, the convergence follows follows from Markov Inequality,

$$\mathbb{P}\{|\widetilde{\sigma}_2^i(s,t) - \sigma_1^i| \geq x\} = \mathbb{P}\left\{|Q_1^*(s) - Q_1^*(t)| \geq \frac{C_1 x}{\rho_1^i}\right\} \leq \frac{2\rho_1^i \mathbb{E}[Q_1]}{C_1 x}.$$

Since $E[Q_1]$ is upper bounded, $\forall x > 0$, the the left hand side of above equation goes to 0 as ρ^i / C_1 goes to 0. This completes the proof.

3.2 Single Output Flow to the $(k+1)$th Node

We now come to generalize the observations on the burstiness of a single flow. Consider the case described by Fig. 2 where the tagged ith input flow $I_i(t)$ traverses k nodes and gives rise to $\widetilde{I}_{k+1}^i(t)$ for node $k+1$.We will use similar approaches as in Section 3.1 to characterize burst size $\widetilde{\sigma}_{k+1}^i$ of $\widetilde{I}_{k+1}^i(t)$.

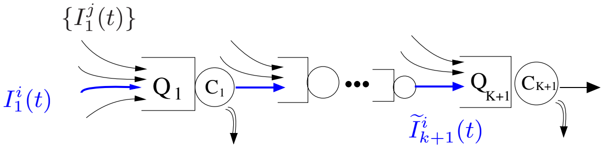


Fig. 2. Scenario S-II

As before, the Virtual Arrival Time Process $v_k^i(t)$ for the ith stream $\widetilde{I}_k^i(t)$ at the kth queue is defined as:

$$v_k^i(t) = \sup_{s \le t}\{s : \widetilde{I}_k^i(s) \le \widetilde{I}_{k+1}^i(t)\} = \sup_{s \le t}\left\{s : s + \frac{Q_k(s)}{C_k} \le t\right\}.$$

and process $v_k^i(t)$ is non-decreasing and right-continuous.

Using the concept of VATP together with the fact that $I_1^i(s,t)$ is regulated, we can obtain Proposition 2. This proposition shows that we can define an envelope process for $\widetilde{I}_{k+1}^i(s,t)$ being stochastically regulated with a burstiness process $\widetilde{\sigma}_{k+1}^i(s,t)$. Proposition 2 can be proved by mathematical induction. Details of the proof is given in [21] due to limited space.

Proposition 2. *Consider an input stream which is characterized by the parameters $(\sigma_1^i, \rho_1^i, \pi_1^i)$ at the first queue. Then its output from the k^{th} queue is stochastically enveloped as:*

$$\widetilde{I}_{k+1}^i(s,t) \le \min\{C_k(t-s), \rho_1^i(t-s) + \widetilde{\sigma}_{k+1}^i(s,t)\},$$

and the burst size process $\widetilde{\sigma}_{k+1}^i(s,t)$ for $\widetilde{I}_{k+1}^i(s,t)$ is defined by

$$\widetilde{\sigma}_{k+1}^i(s,t) = \sigma_1^i + \sum_{m=1}^{k} \frac{\rho_1^i}{C_m}[Q_m(f_m^k(s)) - Q_m(f_m^k(t))]. \tag{9}$$

where

$$f_m^k(t) = v_m^i \circ v_{m+1}^i \circ \cdots \circ v_k^i(t). \tag{10}$$

It is worth noting that $f_m^k(t)$ for the mth queue is a non-decreasing right-continuous process. Moreover, using the knowledge of palm theory [22] as well as mathematical induction techniques, the following lemma shows that the random-time shifting process $Q(f_m^k(t))$ is also a stationary process. Details of the proof is given in [21] due to limited space.

Lemma 2. *If the workload process $\{Q_m(t)\}$ in a work conserving node is a stationary process, for $m = 1, \cdots, k$, then $Q_m^{*k}(t) = Q(f_m^k(t))$ is bounded and $\{Q_m^{*k}(t)\}$ is a stationary process.*

We then obtain the properties of the burstiness of $\widetilde{I}_{k+1}^i(s,t)$ as the following:

Proposition 3. *(Properties of $\widetilde{\sigma}_{k+1}^i(s,t)$)*

The mean value of $\widetilde{\sigma}_{k+1}^i(s,t)$ is given by

$$\mathbb{E}[\widetilde{\sigma}_{k+1}^i(s,t)] = \sigma_1^i. \tag{11}$$

We also have $\widetilde{\sigma}_{k+1}^i(s,t)$ converges to σ_1^i in probability as $\rho_1^i/C_m \to 0$ for $m = 1, \cdots, k$.

Proof. From the stationarity of $Q_j^{*k}(t)$, we have $\mathbb{E}[Q_j^{*k}(t)] = \mathbb{E}[Q_j^{*k}(s)]$ for $j = 1, 2, \cdots, k$, from which equation (11) follows. To show the convergence, from equation (9) and Markov Inequality, we have

$$\mathbb{P}\{|\widetilde{\sigma}_{k+1}^i(s,t) - \sigma_1^i| > x\} \leq \frac{2\rho_1^i}{x} \sum_{m=1}^{k} \frac{\mathbb{E}[Q_m^{*k}]}{C_m}.$$

Therefore, $\widetilde{\sigma}_{k+1}^i(s,t)$ converges to σ_1^i in probability as $\rho_1^i/C_m \to 0$ for $m = 1, \cdots, k$, since $E[Q_m^{*k}(t)]$ is bounded and k is a finite number. This completes the proof.

At the end of this section, we summarize our main result into Theorem 1 for the burstiness behavior of a single flow in acyclic stable networks.

Theorem 1. *In a stable network with work conserving FIFO scheduling at every node with infinite buffer, if all the input flows at the entrance are regulated with leaky-bucket parameters $(\sigma^j, \rho^j, \pi^j)$, then each flow to node k inside the network is still regulated, with a stochastic burst size $\widetilde{\sigma}_k^j(s,t)$. This burst size has the mean value of σ^j and converges exponentially fast to this constant σ^j in probability, as $\rho^j/C_m \to 0$ for $m = 1, \cdots, k-1$.*

Remark 1. This result implies that in a large network, no matter how the flow is multiplexed with others at each node, as long as the average arrival rate of the flow is much smaller than the capacity of the server, its burst size is almost constant. Indeed, it can be a very good approximation when the network carries a large number of flows, as for instance a core network. Furthermore,in our analysis, we consider a heterogeneous network and do not assume independence between input flows. Thus our main result is a general characterization of single regulated flows inside the network.

4 Performance Discussion

We have theoretically shown the convergence of the burstiness of a single flow in Section 3. However, this result is presented in a limit sense: the convergence holds as ρ_i/C_k goes to 0 for each node on an end-to-end path of stream $I_i(t)$. In real network systems, we need to know the scale of the network for the burstiness converges at a reasonable rate as well as how the distribution of its burstiness deviates from its constant mean as the number of end-to-end hops increases. Furthermore, it is more interesting to measure the distortion of performances, such as mean delay and overflow probability, at an internal node along the end-to-end path due to the burstiness behavior of a single flow.

To answer these concerns, in this section, we will simulate the mean delay and overflow probability in a tandem queue scenario as shown in Fig. 2. In particular, we will illustrate how the values of these performance metrics change when the number of input flows increases at each stage and when the number of end-to-end hops increases. These simulation results reflect the convergence behavior of the burstiness of each flow.

Table 1. Simulation Parameters

N_1	N_k^f	C_k	σ_k	ρ_k	π_k
50	49	1	0.1	$0.7/N_1$	1
200	199	1	0.1	$0.7/N_1$	1
400	399	1	0.1	$0.7/N_1$	1

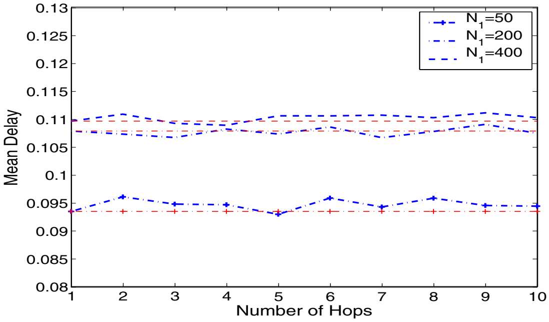
Table 2. Performance Upper bounds with Regulated Inputs

N_1	ED_{INFO}	$P_{Chernoff}$	P_{MD1}
50	0.11433	0.005528	0.013881
200	0.11608	0.012163	0.013881
400	0.11638	0.014004	0.013881

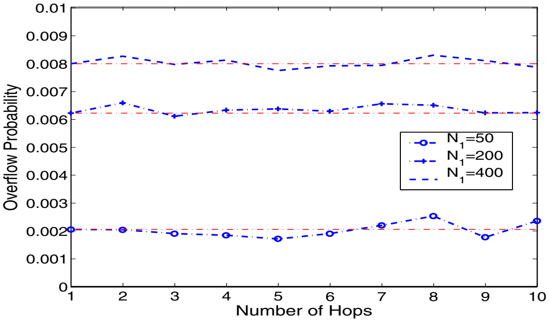
All simulations are carried out on a time-driven fluid simulator. This simulator consists of a number of FIFO queues with infinite buffer size and independent regulated On-Off fresh input streams of chosen parameters. We designed the flows to be homogeneous at each node for simplicity. And we generated independent fresh flows so as to compute the theoretical bounds of the mean delay and overflow probability at each node with available results in the literature ([11, 12]). Note that this independence is required only by computation of the performance metrics. Our results on the burstiness behavior of flows inside the network do not assume independence between flows.

In our numerical simulations, a tandem queue scenario with 10 hops is studied. Parameters about the fresh input flows and the servers are given in Table 1. At each node, we fixed the total load to be 0.7 as the number of sources increases. Therefore, the convergence speed of the burstiness of the ith flow can be measured by the total number of input flows N_k to each node. Furthermore, we set the number of fresh inputs N_k^f, for $k = 2, \cdots, 10$ to be equal to $N_1 - 1$. In this way, input scenarios at the latter 9 nodes are equivalent to replacing one fresh input to node 1 with a regulated flow having stochastic burstiness behavior. We thus can easily observe the evolution of the burstiness behavior of a single flow by comparing values of performance metrics at different nodes.

Numerical results on mean delay at each node are given in Figure 3(a); and those on overflow probability are illustrated in Fig. 3(b). These data are reported for a 95% confidence interval. We omit plotting those intervals in the figures because they are too small to be shown compared to the statitical sample mean. If we let m_k denote the value of a performance metric at node k, and the performance distortion be measured by $(m_k - m_1)/m_1 \times 100\%$, then, as seen from these two figures, when there are more than 200 flows entering each node, performance distortions at each node are negligibly small: around 1% for mean delay and 5% for overflow probability. At the same time, increasing the number of hops of an end-to-end path does not distort the performance at each node dramatically. Indeed, each node along the same path provides similar average delay and overflow probability for arrival flows. All these observations imply that the



(a) Mean Delay at Each Node



(b) Overflow Probability at Each Node with Buffer Threshold B=0.6

Fig. 3. Simulation Results

stochastic burstiness of a single flow converges to its constant mean value at an acceptable speed. In practical applications, we can approximate the stochastic burstiness of a flow inside the network with its mean value, which is the burst size originally obtained at the entrance of the network. Hence, we can obtain the parameters of single flows inside the network and compute the performance of each internal node via single-queue analysis techniques.

At the end of this section, we list in Table 2 the upper bounds of mean delay and overflow probability computed by replacing the stochastic burstiness with its constant mean value. ED_{INFO} is computed from equation (31) in [11]; $P_{Chernoff}$ and P_{MD1} are computed from Chernoff bound and $M/G/1$ queue bound respectively, which are discussed in detail in [12]. It is clear to see that these theoretical upper bounds are still tight to confine the performances of a node *inside* the network, when the inputs to the node are independent of each other as in our scenario.

5 Conclusion

In this paper, we have characterized the burstiness behavior of single traffic flows as they pass through the network. These traffic processes are initially leaky-bucket regulated at the entrance of the network. By considering the internal input traffic as the output flows of some previous queue, we exploit the multiplexing effects from the many sources at the previous node. In particular we have shown that when single flows are small, the multiplexing does not affect their initial burstiness.

This inheritance of burst-size descriptor achieves scalability in designing shapers at the routers of Differentiated Services networks. Shaping algorithm (Leaky-Bucket Algorithm) can be employed only at the edge of the network for each class of flows while core routers can regard each arrival flow keeping its original burst descriptor and do not have to reshape the traffic.

We have also illustrated that statistical performance upper bounds for deterministically regulated flows can be used to estimate the performance of an internal node in scenarios where input flows are independent of each other. Authors in [19] studied by extensive simulation the inter-source cross-correlation under more general settings. They observed that the inter-source dependence benefits queueing performance and independence assumptions between flows can give conservative estimation. Deeper understandings on the dependence between internal flows are indispensable before the establishment of a general methodology for obtaining statistical end-to-end performance estimates.

References

1. D.Anick, D.Mitra, M.Sondhi. Stochastic theory of a data handling system with multiple sources. *Bell System Technical Journal*, vol.61, no.8, pp1871-1894,1982.
2. W. Leland, M. Taqqu, W. Willinger, D. Wilson, On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Trans. on Networking*, Vol. 2, No.1, pp.1-15, Feb. 1994
3. J.W.Roberts. Traffic Theory and the Internet. *IEEE Communications Magazine*, Jan. 2001, pp.94-99.
4. R. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Trans. Inform. Theory*, vol 37, no.1, Jan. 1991 pp. 114-131
5. R. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Trans. Inform. Theory*, vol 37, no.1, Jan. 1991 pp. 132-141.
6. J-Y. Le Boudec, P. Thiran. *Network calculus*, Springer Verlag LNCS 2050, June 2001.
7. C.S. Chang *Performance Guarantees in communication networks*, Springer, 2000.
8. V. Firoiu, J.-Y Le Boudec, D. Towsley, Zhi-Li Zhang, Theories and models for Internet quality of service. *Proceedings of the IEEE*, Vol. 90, Issue 9, Sept. 2002, pp. 1565-1591
9. V. Cholvi, J. Echague, J-Y Leboudec. Worst case burstiness increase due to FIFO multiplexing. In *Proceedings of Performance 2002*, Rome, Italy, Sept. 2002.
10. L. Massoulie, A. Busson. Stochastic majorization of aggregates of leaky bucket constrained traffic streams. Preprint, 2001.

11. F. Guillemin, N. Likhanov, R. Mazumdar, C. Rosenberg. Extremal traffic and bounds on the mean delay of multiplexed regulated traffic streams. In *Proc. of INFOCOM 2002*, N.Y., June 2002, pp. 985-993
12. F. Guillemin, N. Likhanov, R. Mazumdar, C. Rosenberg, and Y. Ying. Buffer overflow bounds for multiplexed regulated traffic streams. In *Proc. ITC 18*, Berlin, Elsevier Science, 2003.
13. M. Vojnovic and J.-Y. Le Boudec. Stochastic analysis of some expedited forwarding networks. In *Proc. of IEEE INFOCOM 2002*, New York, NY, June 2002.
14. O. Yaron, M. Sidi. Performance and stability of communication networks via robust exponential bounds. *IEEE/ACM Trans. Networking*, vol.1 , no.3, June. 1993. pp. 372-385
15. D. Starobinski, M. Sidi. Stochastically bounded burstiness for communication networks, *IEEE Tran. Information Theory*, vol. 46, No.1, Jan. 2000. pp. 206-212.
16. D. Wischik. The output of a switch, or, effective bandwidths for networks. *Queueing Systems* 32, 1999, pp. 383-396
17. D. Eun, N.B.Shroff. Simplification of Network Analysis in Large-Bandwidth Systems. *Proc. of IEEE INFOCOM 2003*, San Francisco, CA, March 2003.
18. D. Abendroth, U. Killat. Intelligent Shaping: Well shaped throughout the entire network. *Proc. of INFOCOM'02*, N.Y. June, 2002.
19. W-c. Lau, San-qi Li. Traffic distortion and inter-source cross-correlation in high-speed integrated networks, *Computer Networks and ISDN Systems* 29(1997) pp811-830
20. T. Konstantopoulos, G. Last. On the dynamics and performance of stochastic fluid systems. *Journal of Applied Probability* 37, pp. 652-667, 2000.
21. Y. Ying, R. Mazumdar, C. Rosenberg, F. Guillemin, Analysis of the burstiness of multiplexed regulated traffic flows in networks. *Technical Report, Purdue University.*
22. F. Baccelli and P. Bremaud. *Elements of Queueing Theory.* Springer-Verlag, 2003.

A Credit-Based Active Queue Management (AQM) Mechanism to Achieve Fairness in the Internet

Gwyn Chatranon[1], Miguel A. Labrador[2], and Sujata Banerjee[3]

[1] University of Pittsburgh
gwync@mail.sis.pitt.edu
[2] University of South Florida
labrador@csee.usf.edu
[3] Hewlett-Packard Laboratories
sujata.banerjee@hp.com

Abstract. Router-based algorithms to address the TCP-friendliness problem have focused on providing fairness to TCP connections by penalizing UDP unresponsive flows. As a result, current schemes have overlooked their effect on streaming applications. In addition, current schemes have not been widely implemented in practice because of their high complexity and their inability to provide fairness when multiple unresponsive flows, packets of different sizes, and bursty traffic are present. All these aspects and scenarios, commonly found in practice, are rarely addressed in the literature. In this paper, we present *Achieving Fairness using a Credit-based mechanism* or AFC, a simple Fair Active Queue Management (FAQM) mechanism that aims to solve the unfairness problems generated under these realistic conditions. Simulation results show that AFC provides smoother transfer rates for unresponsive flows transmitting real-time traffic, handles multiple heavy unresponsive flows better, improves the fairness among TCP connections with different round-trip delays, and achieves good fairness even under bursty traffic and packets of different sizes.

Keywords: Active Queue Management, TCP-friendliness.

1 Introduction

Today, the majority of the Internet traffic such as web traffic, FTP, and email traffic, is carried by the TCP protocol. TCP is widely accepted because of the success of its congestion control mechanism, which enables end hosts to cooperatively adjust their transmission rates according to network conditions, and thus share the available bandwidth fairly among large number of users. Voice and video applications, on the other hand, utilize the UDP protocol. These applications, which are increasing in popularity over the Internet, send *unresponsive traffic* because the UDP protocol provides no end-to-end congestion control. As a result, when TCP and UDP-based applications share the same bottleneck link, the TCP-friendliness problem may arise [1].

Recently, a wide variety of applications and congestion control mechanisms have emerged, which may create an uncooperative environment for end hosts. As a result,

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 930–942, 2005.

leaving the Internet relying purely on end-host mechanisms is a potential risk on network performance and stability. Consequently, router-based mechanisms to address the fairness problem have been widely investigated during the past several years.

One important problem in existing FAQM schemes is that they have primarily focused their attention on providing fairness to TCP connections by penalizing UDP unresponsive flows. As a result, these schemes have overlooked the effect they produce on UDP applications. In addition, current schemes still present some fairness problems due to inaccuracies in the estimation of the number of active flows, innacuracies when multiple unresponsive flows are present, and unfairness when packets are of different sizes and traffic is bursty. All these problems, not analyzed thus far, are the main motivation for the new FAQM scheme proposed here. In addition, CARE [2], which is the newest scheme based on the capture-recapture model, has never been evaluated and compared along with the other schemes.

In this paper, a router-based scheme called AFC (Achieving Fairness using a Credit-based mechanism)[1] is proposed to prevent the unfairness problem using a small amount of per-flow state information while addressing all of these new issues. Through simulations, we compare AFC with CHOKe [4], Stochastic Fair Blue (SFB) [5], BLACK [6], and CARE [2], and show that it provides superior performance. The rest of the paper is structured as follows. Section 3 describes in detail the design goals and algorithms of the AFC scheme. The AFC scheme is then evaluated and compared with other different schemes in Section 4. Finally, conclusions and future work are provided in Section 5.

2 Related Work

Several Fair Active Queue Management (FAQM) schemes have been proposed to provide a fair share of bandwidth to competing flows. These schemes maintain only partial state information, have low computational and space complexity, and need no cooperation from network devices to achieve long-term fairness. In this paper we include CHOKe [4], Stochastic Fair Blue (SFB) [5], BLACK [6], and CARE [2].

CHOKe was introduced in [4] as a mechanism to provide fairness on top of a RED queue [7]. Upon each packet arrival, CHOKe randomly selects a packet from the queue, and drops both packets if they belong to the same flow. High bandwidth flows thus can be controlled as they usually have more packets in the buffer. However, simulations in [4] illustrate that, with CHOKe, a high bandwidth unresponsive flow still can gain much more bandwidth than the fair share. Besides, flows with larger packet sizes gain higher throughput. The only workaround suggested in [8] is to defrag the packets into smaller packets of the same size. Stochastic Fair Blue (SFB) [5] utilizes a bloom filter with multiple levels of hashing to detect high-bandwidth unresponsive flows probabilistically. However, SFB does not include a suggested mechanism to handle unresponsive flows except limiting them to a fixed amount of bandwidth. In practice, setting the bandwidth limit effectively is a difficult task as a fair bandwidth level may be unknown to an SFB router. Besides, if the number of unresponsive traffic is large enough, they

[1] AFC's credit-based mechanism has no relationship with a credit-based flow control in ATM network such as that appeared in [3].

might be hashed into the same locations as responsive traffic, causing a false detection of responsive traffic that would eventually be overpenalized. CARE [2], the most recent scheme, is based on the Capture-Recapture (CR) model that has been widely used to estimate the number of animals in a population and the number of defects in software inspection processes. This model is applied to estimate the flows' arrival rate and the number of active flows, and thus the fair share. Packets from unresponsive flows are dropped according to their proportion of arrival rates that exceed the fair share. The problem of CARE is its core mechanism to estimate the number of active flows. The algorithm is highly complex and presents problems if the intensity of the traffic are highly unequal [9]. Besides, the performance of CARE has never been evaluated along with the other schemes. BLACK, proposed in [6], uses the buffer occupancy fraction as an indicator of the flow's share of the bandwidth. Using a small cache memory, BLACK randomly samples a packet from the queue upon a packet arrival and updates the information in the cache memory to obtain only the high-bandwidth unresponsive flow candidates. BLACK uses a simple memory management adapted from the Least Recently Used (LRU) mechanism [10]. At the end of the sampling period, a buffer occupancy fraction, referred to as *HitFraction*, is estimated and the traffic that consume more than the fair share is penalized according to a dropping probability. In order to estimate the dropping probability p_{drop}, BLACK also estimates the number of active flows, where its inverse value becomes the $FairFraction$.

Even though BLACK has shown superior performance over CHOKe and SFB in [6], all of the fair AQM schemes listed above still contain some limitations. More importantly, these schemes have not yet been analyzed considering real networking situations and issues, such as the unfairness of the schemes due to inaccuracies in the estimation of the number of active flows, unfairness due to traffic with different packet sizes and different round-trip times, the performance of the schemes under bursty traffic conditions, and the fluctuation in the throughput of the flows after passing through the schemes. AFC, which was designed with all these issues in mind, is described next.

3 Achieving Fairness Using a Credit-Based Mechanism (AFC)

This section presents AFC, a new fair AQM scheme that addresses the limitations of the schemes listed in Section 2. AFC modifies BLACK in several ways with newly designed components to make AFC perform better than BLACK and the other schemes under realistic environments. At the same time, AFC inherits and improves over BLACK's important features of low computational and space complexity.

The first modification is related to the estimation of the number of active flows. BLACK presents performance problems because to calculate the number of active flows it assumes that the traffic intensity of the arriving flows is identical. In order to minimize possible estimation errors and to maintain a low level of complexity, in [9] we found that the Direct Bitmap method or any of its variants [11] is the best performing mechanism to estimate the number of active flows thus far. As a result, AFC estimates the number of active flow using the Direct Bitmap mechanism.

Most fair AQM schemes fail to provide fairness when flows send packets of different sizes. For example, CHOKe cannot prevent this problem without breaking a large packet into smaller packets of about the same size, as suggested by the authors in [8]. As CHOKe relies purely on packet matching, between a packet that is sampled from the queue and the arriving packet, a flow that has smaller packet size would be penalized more with the higher probability of matching. This is also the case of BLACK since it computes a flow's buffer fraction based on the number of packets of a particular flow over the number of total sampled packets. To solve this problem, instead of counting the number of packets for the candidate flows in the cache memory, the information is updated with the size of the sampled packet. In addition, the total number of bytes are counted in each period instead of the number of sampled packets. At the end of a sampling period, the *HitFraction* of a flow is calculated by the flow's byte count divided by the number of bytes being sampled.

The third modification is meant to address fairness problems in BLACK during periods of congestion. In BLACK, the $HitFraction$ would represent the average fraction of buffer space used by a particular flow calculated after a sample of m packets if packets were sampled from a virtual queue of size m. However, the way BLACK samples packets does not always resemble the idea of sampling from a virtual queue. BLACK samples packets triggered by packet arrivals, no matter whether the arriving packet will be dropped or enqueued. During congestion, the aggregate arrival rate might be high, and so a high level of packets drop, which is different from a serving rate. While packets are backlogged in the buffer, it is possible that the high sampling rate, as a result of a frequent packet arrival event, may cause the same packet(s) to be sampled more than once. To reduce this possible error, AFC directly collects the $HitFraction$ statistics from the packets that are enqueued and treat them in the same way as sampled packets in BLACK. After the sampling period, the $HitFraction$ of each flow could be determined using the same idea of sampling packets from the virtual queue. It is worth noticing that this new sampling method decreases the complexity of AFC in two ways. First, randomly sampling packets from the queue is no longer required because the information is directly collected from the packet that is enqueued. Second, the update frequency tends to be less because AFC collects the statistics only when there is a packet enqueued excluding those dropped packets, unlike BLACK that the statistics is collected per packet arrival.

An important aspect not considered before is the throughput fluctuation that flows experience after passing through the FAQM mechanism, in particular flows sending data from streaming applications. For example, in BLACK, whenever the $HitFraction$ is higher than the $FairFraction$, incoming packets from flow i are dropped with a certain dropping probability. This means that 1) more incoming packets from flow i might be allowed in, and 2) that already queued packets from flow i might still be in the queue. As a result, after several sampling periods, if the queue is not able to drain the old extra packets and the new packets, the $HitFraction$ may reach the maximum of twice the $FairFraction$, when the dropping probability will be equal to one. After this, the $HitFraction$ will come down to the $FairFraction$ and a new period of fluctuation will begin. AFC introduces a more aggressive dropping function and a *credit-based mechanism* to avoid the cyclical underutilization and overutilization of the bandwidth by unresponsive flows. In order to avoid this fluctuation, once the $HitFraction_i$ is higher than the $FairFraction$, AFC will not allow in more incoming packets from flow i, which implies a dropping probability of one. This dropping probability will continue until the extra packets are drained and the $HitFraction_i$ becomes lower than the $FairFraction$. However, dropping all the packets when a $HitFraction$ becomes higher than a $FairFraction$ might have a problem with responsive flows like TCP, which backs off when their packets are dropped. Once the $HitFraction$ of TCP traffic reaches a $FairFraction$, its incoming packets are dropped causing a back-off period to begin. After a short while, as the queue drains some packets out, the $HitFraction$ becomes lower than the $FairFraction$ once again and incoming packets are allowed to get in. However, the TCP source may still be backing off its data transmission, and thus no or only few packets would arrive at the queue causing the $HitFraction$ to be even lower. Later, after the TCP source expands its congestion window, a burst of packets once again arrives at the queue and the $HitFraction$ eventually reaches the $FairFraction$ again. In other words, the $FairFraction$ becomes an upper limit of the $HitFraction$ for TCP traffic. Since the average $HitFraction$ is less than the $FairFraction$ (See Figure 1 (left)), TCP sources cannot receive their fair share.

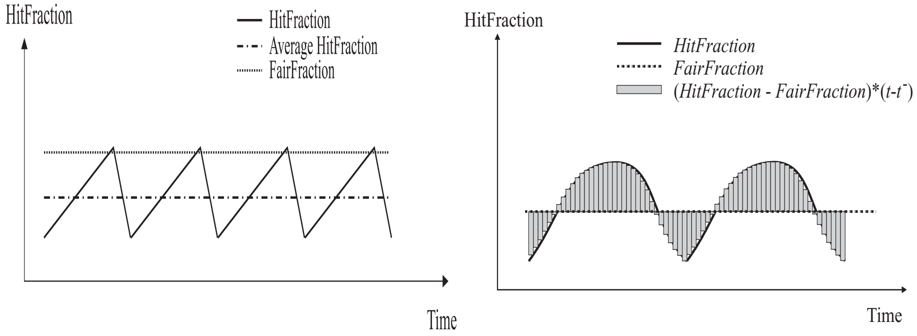


Fig. 1. The problem of aggressive dropping policy to $HitFraction$ (left) and simplified $HitFraction$ behavior of responsive traffic under new AFC dropping policy (right)

The idea of a *credit-based mechanism* in AFC is to solve this problem allowing the $HitFraction_i$ to go beyond the $FairFraction$ if flow i has credit available, so that the average $HitFraction_i$ over time is about the same as the $FairFraction$. Here, a credit is defined as the area under the $HitFraction_i$ curve above or below the $FairFraction$, which is referred to as $\Delta\mathcal{A}$. Precisely, a credit for flow i can be approximated every time a $HitFraction_i$ is updated according to

$$\Delta\mathcal{A}_t = \Delta\mathcal{A}_{t^-} + [(HitFraction_t - FairFraction_t) \times (t - t^-)] \tag{1}$$

as roughly illustrated in Figure 1, where t indicates a current update time and t^- indicates a previous update time. Obviously, the value of $\Delta\mathcal{A}_t$ should be kept as close to zero as possible. However, when a responsive flow is backing off, there is usually not enough packets enqueued to make its $HitFraction$ to raise as much as a $FairFraction$, and thus its $\Delta\mathcal{A}_t$ would become negative. The negative value of $\Delta\mathcal{A}_t$

means that this flow has this amount of credit and AFC will allow the packets of this flow to be enqueued even if its current $HitFraction$ is greater than the $FairFraction$, as long as the flow still has available credit or its $\Delta\mathcal{A}_t$ is still a negative value. This dropping policy is in contrast to the refined dropping function utilized by BLACK in which packets are dropped when a $HitFraction$ is greater than a $FairFraction$ only. By allowing a $HitFraction$ of a flow to be higher than a $FairFraction$ if it has available credit, e.g. from its previous back-off period that causes $\Delta\mathcal{A}_t$ to be negative, an underutilization of an unresponsive flow is prevented.

4 Performance Evaluation

In this section, a comparative evaluation of RED, SFB, CHOKe, BLACK, CARE, and AFC in a number of realistic scenarios is included. We take a simulation approach utilizing the ns-2 [12] simulator and the dumbbell topology shown in Figure 2. The evaluation includes scenarios with single and multiple unresponsive flows and TCP sources with different round-trip times. Because one of the design goals of AFC is to solve the problem of throughput fluctuation, not only the throughput fairness is used as a performance metric but also the instantaneous throughput of CBR traffic over time. Then, more diverse scenarios are conducted to examine fairness and robustness of the fair AQM schemes, including a scenario with traffic with different packet sizes, a scenario with short-lived and bursty traffic, and a scenario when TCP-friendly traffic and TCP traffic are sharing the same bottleneck link.

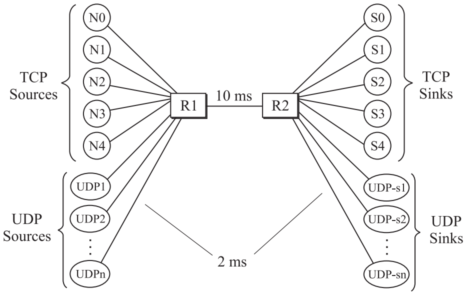


Fig. 2. Simulation topology

Each experiment is run for 200 seconds and is repeated 20 times to calculate 95% confidence intervals. However, for the sake of brevity, we do not present details of the various schemes. The statistics are collected from 50 sec. to 200 sec. The packet size is 1 Kbyte, unless explicitly stated otherwise. SFB is set with a default configuration of two levels of hash functions of 23 bins with double set of hash tables for moving hash functions (total of 46 × 2 bins) using the NS code provided by [13]. The min_{th} and max_{th} threshold settings for RED, CHOKe, BLACK, and AFC are 50 and 150 packets respectively which are the Gentle RED parameters [14]. In the case of CARE, the number of capture occasions (t) is 200, where 50 out of 200 can be used for the estimation of the number of flows, and the probability p_{cap} is 0.04 according to [2].

4.1 Unresponsive Flows

In this section, we present simulation results using single and multiple unresponsive flows. First, a large unresponsive CBR traffic sending data at 5 Mbps shares the 5 Mbps bottleneck link with 100 TCP sources. All the access links are 100 Mbps. Then, we repeat the experiment but using 5 CBR sources. All the queue parameters are unchanged except that CHOKe is now equipped with its *self adjusting mechanism* to handle multiple unresponsive flows. With this mechanism, the region between the minimum threshold (min_{th}) and the maximum threshold (max_{th}) is divided into 8 subregions (k) and the number of packet matching in CHOKe's dropping policy performs $2 \times i$ times per each packet arrival where $i = 1..k$ is the region where the current average queue size is falling into. The Jain's fairness index [15] was utilized to calculate the fairness among the competing flows as follows:

$$f = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{2}$$

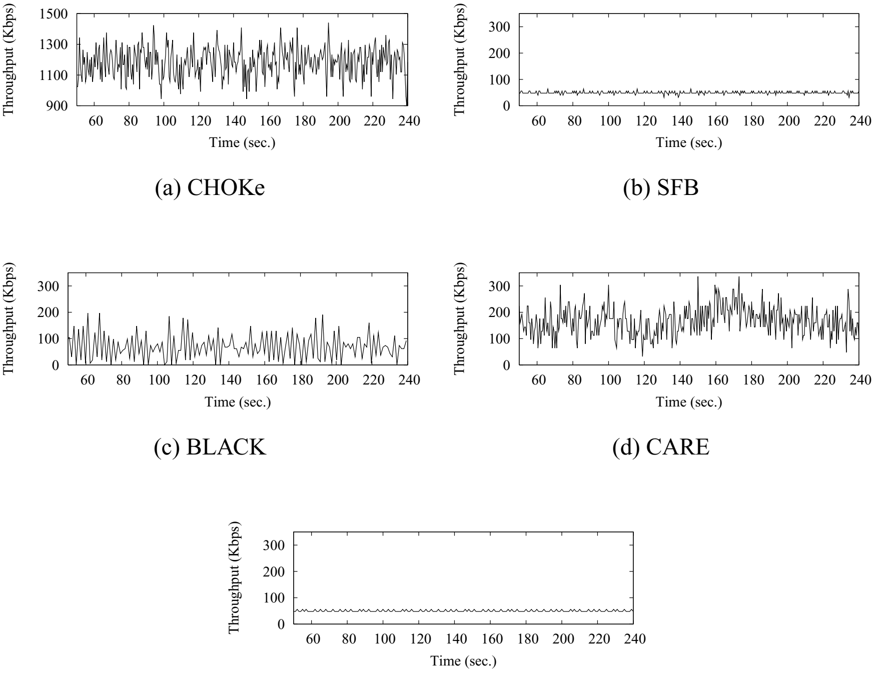
where $0 \leq f \leq 1$, and x_i is the throughput achieved by flow i. The results of the simulations are tabulated in Table 1.

Table 1. Unresponsive flows scenario

Scenario	Single unresponsive flow			Five unresponsive flows		
	Average UDP tput (Kbps)	Average TCP tput (Kbps)	Jain's Fairness index	Average UDP tput (Kbps)	Average TCP tput (Kbps)	Jain's Fairness index
RED	4,374.82	5.046	0.5551	1,000.00	0.000	N/A
CHOKe	1187.301	38.160	0.9842	841.29	5.092	0.1108
SFB, rate limit ≈ twice the fair rate	98.99	49.046	0.9836	95.29	45.285	0.9594
SFB, rate limit ≈ fair rate	49.57	49.538	0.9826	47.84	47.658	0.9611
SFB, rate limit ≈ half the fair rate	24.95	49.78	0.9817	22.94	48.903	0.9593
CARE	172.66	48.307	0.9862	1,000.00	0.000	N/A
BLACK	74.04	49.289	0.9871	65.02	46.378	0.9967
AFC	50.46	49.529	0.9925	48.81	47.609	0.9972

As shown in the table, both RED and CHOKe clearly show their inability to protect responsive TCP flows. TCP traffic are completely shut out in the case of RED and receive only 5.1 Kbps per connection in the case of CHOKe. CARE provides better fairness than RED and CHOKe with one unresponsive flow but its performance deteriorated considerably in the multiple flows case. The trace data from the simulation (not shown) indicates that the precision of CARE's estimation of the number of active flows is altered by the heavy load of multiple unresponsive flows. Although SFB achieves about the same level of fairness than BLACK and AFC, a network operator needs to manually set a rate limit threshold, as SFB has neither knowledge of the fair throughput nor the number of active flows. Besides, without a special per-flow treatment or a separate queue to serve the unresponsive flows, SFB cannot guarantee fairness among unresponsive flows. In the last row of the table, AFC is shown to provide much better fairness than the other schemes in both scenarios. Results not shown here also demonstrate the superiority of AFC in controlling unresponsive traffic even in the case where the arrival rate of the unresponsive flow is as much as twice the bottleneck link rate.

In addition, in terms of the throughput fluctuations for CBR traffic, this phenomenon is greatly reduced under AFC as shown in Figure 3. This is particularly important for streaming applications, which need a smooth transfer rate to perform as users expect. From the figure, it can be seen how CHOKe, BLACK and CARE have a detrimental effect on these applications. Because of the large difference in the UDP throughput over time of CHOKe, we even had to plot the results using a different vertical scale. Only SFB provides a smooth throughput comparable to AFC, while the other schemes result in highly variable throughput. Results not shown here demonstrated that AFC still provided smooth throughput to each of the five CBR sources in the second experiment.



(a) CHOKe

(b) SFB

(c) BLACK

(d) CARE

(e) AFC

Fig. 3. CBR throughput over time after passing through different fair AQM schemes

4.2 Traffic with Different Packet Sizes

An experiment was set up with the same symmetric topology. One hundred TCP flows compete with three CBR flows with an arrival rate of 1 Mbps each. However, these three CBR traffic have different packet sizes, with CBR1 using a packet size of 100 bytes, CBR2 500 bytes, and CBR3 1,000 bytes. The average per-flow throughput of CBR

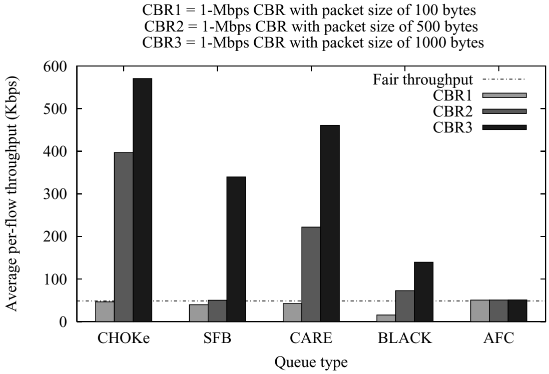


Fig. 4. Average per-flow throughput of CBR traffic with different packet sizes

Table 2. TCP with different round-trip time scenario

	RED	CHOKe	SFB	CARE	BLACK	AFC
Jain's fairness index	0.976	0.972	0.966	0.981	0.991	0.994

traffic under different fair mechanisms are plotted along with a fair share of bandwidth in Figure 4. The figure clearly shows the superior fairness performance of AFC over the other schemes where the per-flow throughput of three CBR with totally different sizes of packets are provided in a fair manner.

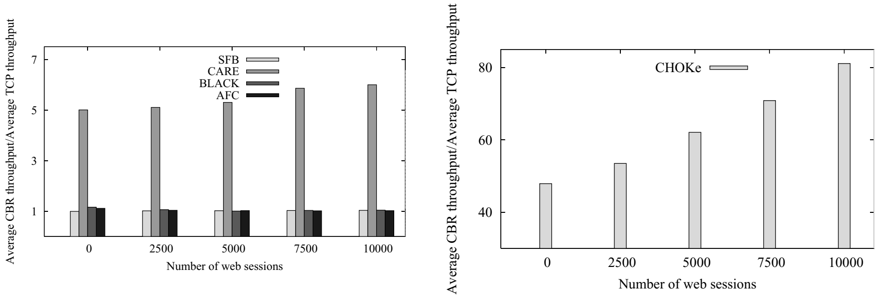
4.3 TCP with Different Round-Trip Times

In this experiment, 200 TCP traffic are randomly originated from nodes N0, N1, N2, N3 or N4 and linked to the randomly selected sink nodes S0, S1, S2, S3 and S4 with an asymmetric topology where the link from N0 to R1 and the link from R2 to S0 in Figure 2 have 1 ms of propagation delay each, N1-R1 and R2-S1 5 ms, N2-R1 and R2-S2 10 ms, N3-R1 and R2-S3 15 ms, and N4-R1 and R2-S4 20 ms. The cache size of BLACK and AFC is 46 which is the same as SFB, or only a quarter of the number of these long-lived TCP traffic. The results presented in Table 2 clearly show that AFC provides the best fairness performance among TCP connections with different round-trip times.

4.4 Effect of Short-Lived Traffic

In all of the previous experiments, the fairness performance is evaluated under the ideal scenarios where all of the sources transmit unlimited amount of traffic. In this section, the schemes are evaluated using two types of short-lived traffic: 1) low-bandwidth short-lived traffic such as web traffic, and 2) high-bandwidth short-lived traffic such as malicious traffic that aims at saturating a link by escaping a fair AQM mechanism.

Low-Bandwidth Short-Lived Traffic. In this part, experiments are conducted to evaluate how fair the AQM schemes are when there are different loads of web traffic in the background. With the same topology, five CBR sources of 2.5 Mbps each and 100 TCP sources are sharing the same bottleneck link along with w sessions of web traffic. Each web session contains a default parameter recommended in the ns-2 script [12] where the traffic model is based on [16]. A Pareto distribution is used for flow lengths of web traffic where the average number of packets per flow is 15 with a shape parameter of 1.2. The starting time of each of the w web sessions is randomly set during 250 seconds of simulation time. With a fixed simulation time, a large w implies a high web traffic load scenario or more web traffic that would arrive at the queue than with a small w. In this experiments, the number of web sessions w are set to 0, 2, 500, 5,000, 7,500 and 10,000 sessions to be generated during this 250 seconds of simulation time.



(a) SFB, CARE, BLACK, and AFC

(b) CHOKe

Fig. 5. Average per-flow CBR throughput over average per-flow TCP throughput at different background web traffic loads

Figure 5 shows the average per-flow CBR throughput over the average per-flow TCP throughput. Ideally, the ratio should be close to one to ensure fairness among different connections. The figure shows that SFB (with well-tuned rate limit), CARE, BLACK, and AFC achieve their fairness performance with minimal interference from different web traffic loads. The average CBR throughput under CARE is higher than SFB, BLACK, and AFC because of the high arrival rates that causes an underestimation of the number of active flows. However, CHOKe shows a different result. When no background web traffic is presented, the average per-flow CBR throughput is about 50 times the average long-lived TCP throughput, and the average CBR throughput gets significantly higher than the average TCP throughput as the number of web sessions increases. The rationale behind this poor performance of CHOKe is that when there are more packets from the web traffic in the queue, fewer packets from the same CBR connection are in the queue, which leads to less chance of a packet matching and less control of unresponsive traffic.

High-Bandwidth Short-Lived Traffic or Bursty Traffic. Although the fair AQM mechanisms achieve long-term fairness in different scenarios, no evaluation has included high-bandwidth short-lived misbehaving traffic. Fair AQM schemes that do not maintain per-flow state information might not have enough long term information about the average arrival rate of these flows and therefore might not be able to achieve fairness. Some AQM schemes need some amount of time to collect information before identifying and controlling misbehaving traffic, and may erase that information after a short while. Providing long-term fairness under high bandwidth short-lived traffic may not be possible for these lightweight fair AQM schemes, and a key question here is how well these schemes can detect and control these types of traffic.

A series of experiment are set up with misbehaving traffic represented by a high-bandwidth short-lived UDP traffic with different ON and OFF periods, which are drawn from an exponential distribution. Two sets of experiments are set for two peak rates of 1 Mbps and 10 Mbps. Again, 100 long-lived TCP sources are passing through the same bottleneck link of 10 Mbps. The results are summarized in Table 3.

Table 3. Scenario with 1 Mbps and 10 Mbps peak rate bursty traffic

	1 Mbps peak rate bursty traffic scenario				10 Mbps peak rate bursty traffic scenario			
No. of UDP	5 flows		15 flows		5 flows		15 flows	
	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)
RED	456.9	77.2	405.0	39.4	1698.9	9.8	666.3	0.0
CHOKe	385.2	80.8	364.4	45.5	893.9	55.3	653.3	1.9
SFB	321.1	84.0	375.6	43.8	86.1	95.7	115.6	82.7
BLACK	257.1	87.2	264.2	60.5	595.7	68.9	385.7	37.2
CARE	213.0	89.4	264.8	60.4	1098.7	44.4	666.4	0.0
AFC	265.7	86.7	245.1	63.4	231.4	87.7	191.1	68.4

For bursty traffic with 1 Mbps peak rate case and 5 CBR flows, although all of the schemes leave the bandwidth to each bursty traffic at least 1.8 - 3 times the fair share, all of the schemes still provide some level of protection to TCP connections. However, with 15 CBR bursty sources, RED clearly provides the least fairness, as all bursty traffic altogether take up to 60% of the total bandwidth and leave each TCP connection with a bandwidth equal to only about half of the fair share. Here, CARE estimates the number of active flows much better as the arrival rate of the bursty traffic is smaller. The result of SFB is different because SFB needs to know a rate limiting threshold in advance, which should be manually configured. Using the same settings that achieve good fairness in the 10 Mbps peak rate case, SFB turns to provide poorer fairness performance in this 1 Mbps peak rate case. This result shows that although SFB could detect unresponsive traffic, it cannot use the same settings to provide fairness for different scenarios.

For bursty traffic with 10-Mbps peak rate, the RED mechanism cannot protect TCP traffic in all the scenarios. CHOKe and CARE could provide some protection when there are 5 bursty flows, but cannot prevent 15 bursty flows from grasping almost all of the bandwidth. It is expected for CHOKe as the results in the previous experiments show that CHOKe does worse in providing fairness under a higher number of unresponsive flows. CARE, however, is a little worse than CHOKe because of its inability to estimate the number of active flows correctly with unresponsive traffic with high arrival rates. BLACK provides some level of protection as it can reserve about 60% - 70% of the fair throughput to each TCP connection on average when there are 5 bursty traffic. A trace file indicates that BLACK's $HitFraction$ mechanism, that randomly samples packets from the queue, does not perform as well as when the unresponsive traffic is non-bursty. On the other hand, the direct counting of $HitFraction$ of AFC provides better fairness, as each TCP connection gains more than 92% of the fair share. However, both BLACK and AFC performance are degraded when there are 15 bursty traffic coming to the queue because of the cache size of only 20. In this case, bursty traffic does have an impact on the mechanism of BLACK and AFC, as that traffic could be replaced easily during the OFF period of the traffic. Here, BLACK turns to be largely distorted, while AFC is still far better as each TCP connection gains 72% - 90% of the fair share. On the other hand, SFB performs well in these scenarios, but with a fine tuning of rate limiting given that the arrival rate of bursty traffic is known in advance.

5 Conclusion

This paper proposes AFC, a novel fair AQM scheme that outperforms current schemes in well-known as well as new and more realistic scenarios not utilized before. Using simulations, we evaluate and compare AFC with RED, SFB, CARE and BLACK including not only a single unresponsive flow but also multiple unresponsive flows, TCP connections with different RTTs, low and high bandwidth bursty background traffic, and different packet sizes. We show that AFC provides the best fairness performance in all these cases with low complexity and memory requirements. In addition, AFC's credit-based mechanism avoids oscillations and underutilization of responsive flows and provides smooth transfer rates to unresponsive flows.

References

1. S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.
2. M. Chan and M. Hamdi, "An Active Queue Management Scheme Based on a Capture-Recaptured Model," *IEEE J. Select. Areas Commun.*, vol. 21, no. 4, May 2003.
3. H. T. Kung, T. Blackwell, and A. Chapman, "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing," in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 101–114.
4. R. Pan, B. Prabhakar, and K. Psounis, "CHOKe - A Stateless Active Queue Management Scheme For Approximating Fair Bandwidth Allocation," in *Proc. IEEE INFOCOM*, April 2000, pp. 942–951.
5. W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," in *Proc. IEEE INFOCOM*, April 2001, pp. 1520–1529.
6. G. Chatranon, M. A. Labrador, and S. Banerjee, "BLACK: Detection and Preferential Dropping of High Bandwidth Unresponsvie Flows," in *Proc. IEEE ICC*, May 2003, pp. 664–668.
7. S. Floyd and V. Jacobson, "Random Early Detection Gatewas for Congestion Avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
8. K. Psounis, R. Pan, and B. Prabhakar, "Approximate fair dropping for variable-length packets," *IEEE Micro*, vol. 21, no. 1, January/February 2001.
9. G. Chatranon, M. A. Labrador, and S. Banerjee, "A Survey of TCP-Friendly Router-based AQM Schemes," *Computer Communications*, vol. 27, no. 15, pp. 1424–1440, August 2004.

10. I. Kim, “Analyzing Network Traces To Identify Long-Term High Rate Flows,” Master’s thesis, Texas A&M Univ., May 2001.
11. C. Estan, G. Varghese, and M. Fisk, “Counting the Number of Active Flows on a High Speed Link,” *ACM SIGCOMM Computer Communication Review*, vol. 32, July 2002.
12. “NS Network Simulator,” http://www.isi.edu/nsnam/ns/.
13. “The Network Simulator: Contributed Code,” http://www.isi.edu/nsnam/ns/ns-contributed.html.
14. S. Floyd, “Recommendation on using the ”gentle_” variant of RED,” 1999, http://www.icir.org/floyd/red/gentle.html.
15. R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.
16. A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger, “Dynamics of IP traffic: A study of the role of variability and the impact of control,” in *Proc. ACM SIGCOMM*, Sept. 1999.

Optimizing End-to-End Throughput for Data Transfers on an Overlay-TCP Path[⋆]

Pradnya Karbhari, Mostafa Ammar, and Ellen Zegura

College of Computing, Georgia Institute of Technology, Atlanta, GA-30332
{pradnya, ammar, ewz}@cc.gatech.edu

Abstract. We consider an overlay network where traffic on an overlay hop is carried in one or more TCP connections established between the overlay nodes at the ends of the hop. We are interested in maximizing the throughput of data carried by this type of overlay network. To that end, we focus on a single path in such a network and investigate how one can determine the number of TCP connections on each overlay hop so as to maximize the throughput of the data carried end-to-end on that path. We first show that having more than one TCP connection on some overlay hops can indeed increase the throughput on overlay paths. We then propose the *Adaptive Overlay-TCP Provisioning* approach, that, based on the path state, dynamically introduces and removes TCP connections on individual overlay hops to maximize throughput while minimizing the overhead of extraneous connections. We consider two schemes to assess the path state, the *intermediate buffer occupancy scheme* and the *isolated rate probing scheme*, and evaluate them experimentally on a set of Planetlab nodes. We show that these schemes can significantly improve the end-to-end throughput with very little overhead.

Keywords: Overlay networks, parallel TCP connections.

1 Introduction

Overlay networks have been proposed for a variety of applications such as content distribution [1], multicast [2], quality of service [3] and multimedia streaming [4]. Ideally, these multiple overlay networks should coexist without harming each other. In particular, it is important that coexisting overlay networks share native network link bandwidths in a fair manner, a problem identified in [5]. One way to achieve this is to carry data on overlay hops using one or more TCP connections between overlay nodes at the ends of the hops. We refer to such overlay networks as *Overlay-TCP networks*[1]. Such networks have the desirable property that they ensure that multiple overlays will share native link bandwidths fairly, and that

[⋆] This work is supported by NSF grant ANI-0240485.

[1] In this work, we consider overlays that provide a permanent and tunable infrastructure to support applications. We do not look at overlays resulting from peer-to-peer systems that have higher churn rate amongst nodes and hence amongst overlay links.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 943–955, 2005.

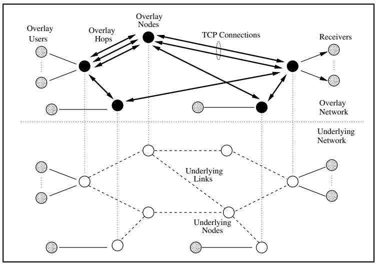


Fig. 1. Overlay Network

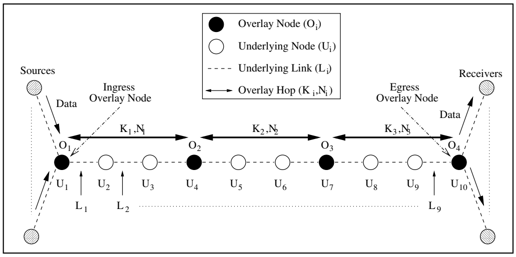


Fig. 2. Model for Overlay Path

the Internet will not suffer due to lack of congestion control in the network. Figure 1 shows an example of an Overlay-TCP network.

Our ultimate goal is to maximize the throughput of data carried on Overlay-TCP networks. To that end, in this paper, we focus on the problem of maximizing the throughput on a path in such an overlay network. We assume the existence of a separate routing infrastructure that determines the overlay path from a sender S to a receiver R. The main design parameter in this work is the number of TCP connections on each overlay hop in the path. To understand the need for such a design, we define the *isolated rate* of an overlay hop as the throughput that a single long-lived continuously backlogged TCP connection would achieve on that hop. Now consider an Overlay-TCP path in which a single TCP connection is used on each hop. Clearly, the isolated rates on the overlay hops are usually not equal, and vary over time. The end-to-end throughput on this path is limited by the minimum of the isolated rates on the individual overlay hops.

This end-to-end throughput can be improved by introducing multiple parallel connections on slower overlay hops to increase their rate to match that of faster overlay hops. This might be considered an aggressive measure, and unfair to connections competing on the bottleneck native link. However, the addition of a bound on the number of parallel connections can provide effective control on this unfairness and equalize treatment among multiple overlays. Also, we are using TCP connections, which are congestion responsive. Finally, although we intend to use parallel connections on overlay hops, we aim to maximize the throughput using as few connections as possible.

The improvement in end-to-end throughput achieved with the use of multiple parallel connections comes at the expense of packet reordering at the receiver. We observed experimentally that the reordering index[2] increases with increasing number of parallel connections between two nodes, especially for overlay paths of more than 2 hops. The receiver has to buffer and reorder all data sent over multiple parallel connections, thus adding some complexity. Thus, packet reordering is an added incentive to keep the number of parallel connections to a minimum.

[2] We define the *packet reordering index* as the root mean square error of the received packet order compared to the initial sending order.

We first show that using more than one connection on some overlay hops does indeed increase the throughput on overlay paths. We then propose the *Adaptive Overlay-TCP Provisioning* approach using two schemes (*intermediate buffer occupancy scheme* and *isolated rate probing scheme*) that assess the path state and dynamically introduce and remove TCP connections on individual overlay hops. We evaluate these schemes on a set of PlanetLab [6] nodes[3] and show that our schemes significantly increase the end-to-end throughput with as few extraneous connections as possible. We discuss the appropriate parameters to be used so as to be less aggressive while maximizing the end-to-end throughput.

Analysis of TCP connections on consecutive overlay hops has been undertaken in various contexts in quite a few papers [8, 9, 10, 11]. All these papers employ schemes such as TCP backpressure, packet dropping, and explicit congestion notification to throttle connections to the rate of the slowest hop. None of these consider the option of using multiple TCP connections on overlay hops. Amir et al. [11] mention the possibility of multiple connections on each hop, but do not explore it further. In the context of reliable multicast, a number of studies [2, 12, 13] show that the end-to-end throughput of an overlay path, and consequently the whole multicast tree is limited by the overlay hop with the minimum throughput. The effect of using end-to-end parallel TCP connections for transferring data on a single path, or from multiple servers on multiple paths has been studied earlier [14,15,16], but not on overlay paths of two or more hops.

The outline of the paper is as follows. We formulate the problem in Section 2 and present a case study of a 2-hop overlay path in Section 3. We discuss the proposed Adaptive Overlay-TCP Provisioning architecture in Section 4, outline the proposed schemes in Section 5, and evaluate their performance in Section 6. In Section 7, we summarize the conclusions and discuss future work.

2 Problem Statement

Our model for an overlay path is shown in Figure 2. The parameters and variables for the overlay path are listed in Table 2. Consider a path of m nodes, each node denoted by U_i in the underlying network. These nodes are connected by $m-1$ links, each denoted by L_i. Let n of the m underlying nodes function as overlay nodes, where $n \leq m$. The overlay nodes, denoted by O_i, form an overlay path of $n-1$ hops, each denoted by K_i, with N_i TCP connections on each hop. The overlay path comprises one or more links and nodes in the underlying network.

We consider one-way data transfer on this path. A set of sources send data to a set of receivers. The senders and receivers need not be part of the overlay network. Sources send data to an ingress overlay node, which detects the transfer, say by looking at headers, and then forwards it over the Overlay-TCP network, as

[3] We realize that PlanetLab, like any other experimental infrastructure, is not completely representative of the real Internet [7]. However, it is currently the best infrastructure available to researchers to evaluate their work in a relatively realistic scenario on the Internet.

Table 1. Model of an Overlay Path

System Parameters	
m	Number of underlying nodes on the path
n	Number of overlay nodes on the path
System Variables	
U_i	i^{th} underlying node on the path, $i = 1..m$
O_i	i^{th} overlay node on the path, $i = 1..n$
L_i	i^{th} underlying link on the path, $i = 1..m-1$
K_i	i^{th} overlay hop on the path, $i = 1..n-1$
N_i	Number of TCP connections on i^{th} overlay hop on the path, $i = 1..n-1$
R_i	Average isolated rate on i^{th} overlay hop on the path, $i = 1..n-1$

suggested by the proxy mechanism in [17]. The egress overlay node demultiplexes the data and delivers it to the appropriate receiver. We assume that the aggregate incoming data at the ingress overlay node keeps the connection continuously backlogged. The overlay path is assumed to carry background traffic.

Given such an overlay path, with varying background traffic, our goal in this work is to dynamically determine and provision the number of TCP connections necessary on each overlay hop in order to maximize the end-to-end throughput. We refer to this approach as Adaptive Overlay-TCP Provisioning.

3 Case Study: 2-Hop Overlay Path

We start by studying the behavior of a 2-hop overlay path with a single connection on each hop. At the intermediate overlay node, the socket buffer for the incoming TCP connection forwards the data packets to an application layer buffer. This buffer then forwards the packets to the socket buffer for the outgoing TCP connection. See Figure 4 for more details.

Recall that we define the *isolated rate* of an overlay hop as the throughput that a single long-lived continuously backlogged TCP connection would achieve on that overlay hop. The isolated rates of the incoming hop (R_1) and the outgoing hop (R_2) at any overlay node will generally be unequal. We define the *degree of mismatch, M* ($M \geq 1$) at an overlay node as $M = \frac{max(R_1,R_2)}{min(R_1,R_2)}$.

1) If $R_1 < R_2$, data comes in at the intermediate node at a rate slower than what can be forwarded on the outgoing hop. The outgoing connection does not have enough data to operate at its isolated rate (R_2) and is limited by the data rate of the incoming connection. Also, since it is not continuously backlogged any more, it might repeatedly go into slow start and get an even lower throughput.

2) If$R_1 > R_2$, data comes in at the intermediate node at a rate faster than what the outgoing connection can keep up with. Although the outgoing connection can now operate at the maximum rate (R_2), the intermediate node has to

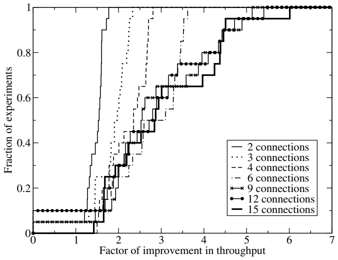


Fig. 3. Improvement in throughput using multiple connections

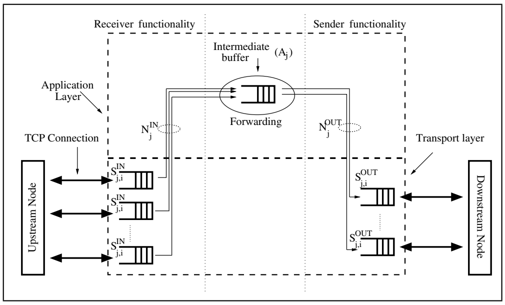


Fig. 4. Model for Intermediate Overlay Node O_j

buffer the excess data. Once the buffers at the intermediate node are full, the incoming connection has to throttle back. It can send more data only after the buffers have space to hold it. The incoming connection is now bounded by the data rate of the outgoing connection. Also, now the incoming connection might go into repeated slow-starts, thus lowering the throughput even more.

3) If $R_1 = R_2$, the average connection rates over some time period T are equal. However, at any instant, each TCP connection will be in the slow start or congestion avoidance phase. These phases need to be synchronized in order to get the maximum effective throughput.

Effectively, the end-to-end throughput on a 2-hop overlay path with a single TCP connection on each hop is bounded by the throughput of the slower connection on the path. In an attempt to improve the end-to-end throughput, we evaluate the effect of using multiple TCP connections on the slower overlay hop.

We set up 2-hop overlay paths on a set of Planetlab [6] nodes and determined the isolated rates on each hop by performing a 50 MB transfer on a TCP connection between consecutive nodes. We then started data transfers with one TCP connection on the faster hop and multiple (2-15) connections on the slower hop. Figure 3 shows a sample of results obtained on an overlay path where nodes at Duke, Georgia Tech (GT) and U-Arizona were used as ingress, intermediate and egress overlay nodes respectively. On this path, the isolated rate from Duke to GT was higher than that between GT and U-Arizona ($R_1 > R_2$). The figure plots a CDF of the factor of improvement, defined as the ratio of the end-to-end throughput using multiple connections on an overlay hop, to that using a single connection on each hop. We see that the factor of improvement increases with the number of downstream connections. However, the effect of adding new connections decreases as the number of connections increase. We observed similar results for the $R_1 < R_2$ case, with multiple connections on the incoming hop.

We conclude from these experiments that adding multiple TCP connections on a slower hop does improve end-to-end throughput. However, beyond a threshold, adding new connections does not have a significant effect on the throughput.

Table 2. Adaptive Overlay-TCP Provisioning: Parameters and Variables

Parameters	
N_{max}	Maximum number of TCP connections between consecutive overlay nodes
A_j	Size of application layer buffer
$S_{j,i}^{IN}$	Size of socket buffer for each incoming connection
$S_{j,i}^{OUT}$	Size of socket buffer for each outgoing connection
Variables at Overlay Node O_j	
M_j	Degree of mismatch between upstream and downstream isolated rates
N_j^{IN}	Current number of incoming TCP connections
N_j^{OUT}	Current number of outgoing TCP connections
$B_{j,i}^{IN}$	Current occupancy of socket buffer for each incoming connection
$B_{j,i}^{OUT}$	Current occupancy of socket buffer for each outgoing connection

4 The Adaptive Overlay-TCP Provisioning Architecture

Generalizing the above discussion to an n-hop overlay path, the effective end-to-end throughput of data transfers on this path, with a single TCP connection on each hop, will be limited by the slowest TCP connection on the path. We can improve the throughput by using multiple TCP connections on slower hops.

The architecture for a system to provide *Adaptive Overlay-TCP Provisioning* is based on the model of an overlay node O_j, shown in Figure 4. An overlay node, denoted by the dashed box, is split into the Application and Transport layers. A pair of consecutive overlay nodes can communicate over a maximum of N_{max} TCP connections. N_{max} is a system-wide parameter. At any point of time, the node O_j has N_j^{IN} active TCP connections with the upstream node, and N_j^{OUT} active TCP connections with the downstream node. These connections read or write data into transport layer socket buffers, each with a maximum capacity of $S_{j,i}^{IN}$ and $S_{j,i}^{OUT}$ for incoming and outgoing TCP connections respectively. All incoming connections forward packets from the socket buffers to the application buffer, of capacity A_j. Effectively, each node has a buffering capacity of $\sum_{i=1}^{N_{max}} S_{j,i}^{IN} + \sum_{i=1}^{N_{max}} S_{j,i}^{OUT} + A_j$. For each node j, $B_{j,i}^{IN}$ and $B_{j,i}^{OUT}$ denote the instantaneous incoming and outgoing socket buffer occupancy for connection i.

The proposed architecture consists of the following three components:

1) *Network Condition Evaluation Module:* This module evaluates network conditions under which the TCP connections operate by measuring one of two quantities. A direct approach is to periodically probe the isolated rates on each hop by performing a data transfer. We can also indirectly measure the relative isolated rates by measuring the buffer occupancy of incoming TCP connections.

2) *The Decision Algorithm:* Based on the measured quantities, the decision algorithm decides whether multiple connections are required on any hop on the overlay path. If yes, it also gives either the number of connections

```
if ($R_1 > R_2$) // faster incoming link
    $M = R_1/R_2$ // calculate degree of mismatch
    if ($M < N_{max}$)
        $N_j^{OUT} = (int)M$ // assign higher number of outgoing connections
    elseif ($M \geq N_{max}$)
        $N_j^{OUT} = N_{max}$ // do not exceed maximum connection count
    endif
elseif ($R_2 > R_1$) // faster outgoing link
    $M = R_2/R_1$ // calculate degree of mismatch
    if ($M < N_{max}$)
        $N_j^{IN} = (int)M$ // assign higher number of incoming connections
    elseif ($M \geq N_{max}$)
        $N_j^{IN} = N_{max}$ // do not exceed maximum connection count
    endif
endif
```

Fig. 5. Decision Algorithm using Isolated Rates

required on each hop, or a decision to increase or decrease the number of connections.

3) Connection Setup and Maintenance: The varying number of upstream and downstream connections can be implemented in two ways. First, we can keep N_{max} connections active at all times. At any point of time, we use N_j^{IN} and N_j^{OUT} connections, as computed by the decision algorithm, to send data. Another option is to start a new connection and tear down an existing one every time the decision algorithm alters either N_j^{IN} or N_j^{OUT}. Due to the connection setup overhead associated with this option, we use the first option in our solution.

The number of connections required on each hop might change over the course of the data transfer due to changing network conditions. Hence, in our schemes, we periodically evaluate the network conditions and repeat the decision process.

5 Proposed Schemes

Based on the two quantities measured by the network condition evaluation module, we propose two schemes and corresponding decision algorithms to determine the number of connections required on each overlay hop.

5.1 Direct Measurement Approach: Isolated Rate Probing Scheme

In this scheme, we directly measure the isolated rate on each overlay hop by periodically performing a 250 KB transfer on a TCP connection on that hop. Although this scheme gives a relatively accurate estimate of the isolated rates, it places additional load on the path of the data transfer. We can use the lightweight alternative of measuring loss probability and round-trip time for each hop, and estimating the TCP throughput using the TCP equation [18] to give an estimate of the isolated rates. In this work we do not implement this alternative.

```
if ($\hat{B} > \beta * B$) // overfull buffer
    if ($N_j^{IN} > N_j^{OUT}$) // too many incoming connections
        $N_j^{IN} --$
    elseif ($N_j^{IN} \leq N_j^{OUT}$) // not enough outgoing connections
        if ($N_j^{OUT} < N_{max}$)
            $N_j^{OUT} ++$
    endif
elseif ($\hat{B} < \alpha * B$) // underfull buffer
    if ($N_j^{IN} \geq N_j^{OUT}$) // not enough incoming connections
        if ($N_j^{IN} < N_{max}$)
            $N_j^{IN} ++$
    elseif ($N_j^{IN} < N_j^{OUT}$) // too many outgoing connections
        $N_j^{OUT} --$
    endif
else
    Do nothing
endif
```

Fig. 6. Decision Algorithm using Buffer Estimates

2-*Hop Overlay Path:* Figure 5 shows the decision algorithm at overlay node O_j to determine the location and number of connections on a 2-hop overlay path, with isolated rates R_1 and R_2 on incoming and outgoing hops respectively. If R_1 is greater, the algorithm computes the degree of mismatch (M) in Line 2, which is the required number of outgoing connections (Line 4). Lines 8-14 handle the case when R_2 is greater. Lines 5 and 12 ensure that N_{max} is not exceeded.

Multi-hop Overlay Path: We now extend the algorithm to the multihop case. We compute M_i^{max}, the maximum degree of mismatch for each hop along the overlay path as the ratio of the maximum isolated rate (R_{max}) on the path, to the isolated rate (R_i) of that hop. This ratio gives the number of connections required on that hop. Taking N_{max} into account, the number of downstream connections required at each node is given by $N_i^{OUT} = min(\frac{R_{max}}{R_i}, N_{max})$. To implement this algorithm, the ingress node sends its downstream isolated rate to its downstream node, which appends its own downstream isolated rate and forwards to the next node and so on. Once the vector reaches the egress node, it computes N_i^{OUT} for each hop and sends the vector back along the path.

5.2 Intermediate Buffer Occupancy Scheme

The above scheme can incur a significant overhead for determining isolated rates and signaling among overlay nodes. We now describe a scheme based on local observation of an overlay node's buffers. To illustrate this scheme, consider a 2-hop overlay path where the isolated rate on the incoming hop is greater than that on the outgoing hop ($R_1 > R_2$). In this case, the buffer will be relatively full most of the time, whereas, if $R_1 < R_2$, the buffer will be relatively empty. We validated this by observing the incoming socket buffer occupancy over time for several 2-hop overlay paths on a set of PlanetLab nodes.

For the general multi-hop case, Figure 6 shows the decision algorithm at an overlay node to determine whether it needs to either increase or decrease the number of connections on the incoming or outgoing hop. This algorithm operates at each overlay node independently. It uses a buffer occupancy estimator, $\hat{B}$, based on an exponentially weighted moving average of periodic samples of instantaneous buffer occupancy as follows, $\hat{B} = \gamma * \hat{B} + (1 - \gamma) * \hat{B}_{sample}$, where $0 < \gamma < 1$. Let α and β be the low and high watermarks respectively, below which we consider the buffer to be underfull, and above which we consider it to be overfull. These parameters are the fraction of the total buffer space available at the receiving end of the incoming TCP connection at the intermediate node.

If the buffer is overfull, the algorithm checks whether the number of incoming (N_j^{IN}) or outgoing connections (N_j^{OUT}) is higher. If N_j^{IN} is too high (Line 2), the outgoing link might be unable to keep up, hence we decrease N_j^{IN}. If N_j^{IN} is less than N_j^{OUT} (Line 4), the outgoing connections cannot keep up even with very few incoming incoming connections, hence we increase N_j^{OUT}. Line 5 ensures that this number does not exceed N_{max} at any point of time. Lines 8-14 handle the case when the buffer is underfull. If the current buffer occupancy is within the acceptable range ($\alpha * B \leq \hat{B} \leq \beta * B$), we do not take any action.

At any overlay node, a change in N_j^{OUT} is implemented immediately. However, a change in N_j^{IN} is communicated to the upstream node, which changes its N_j^{OUT} if its own decision algorithm concurs. The ingress node implements any decision by its downstream node, as it does not do any buffer estimation itself.

With this scheme, although we can measure network conditions without introducing additional traffic and do the estimation while the data transfer is in progress, we cannot determine exact values of the isolated rates, and hence we do not know the exact number of connections required. We need to continuously monitor and increase or decrease the number of connections by one every time.

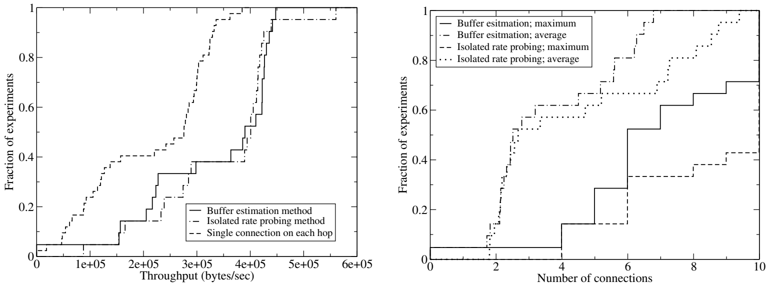
6 Performance Evaluation

In this section, we present a set of experimental results, based on experiments on Planetlab [6] nodes. We discuss the measurement methodology, the parameters for the schemes, and then the performance results.

6.1 Measurement Methodology

We experimented with multiple overlay paths of 2-5 hops, by selecting 3-6 overlay nodes for each path from the set of available Planetlab [6] nodes. For the isolated rate probing scheme, we first started a set of control processes that performed a 250 KB transfer every 30 seconds to estimate the current value of R_i. We then started parallel transfers of 100 MB each between the following nodes.

1) Direct TCP transfer between all sets of consecutive overlay nodes. These are used to monitor the isolated rates on each overlay hop.
2) Overlay-TCP transfer from ingress overlay node to egress overlay node with a single TCP connection on each overlay hop. We compare the throughput of data transfers using our schemes to this benchmark case, which would be the one used in an Overlay-TCP path without multiple connections.



(a) End-to-End Throughput (b) Number of Connections

Fig. 7. 2-hop overlay path

3) Overlay-TCP transfer from ingress overlay node to egress overlay node using our Adaptive Overlay-TCP Provisioning schemes to maintain the appropriate number of parallel connections between overlay nodes.

6.2 Parameters

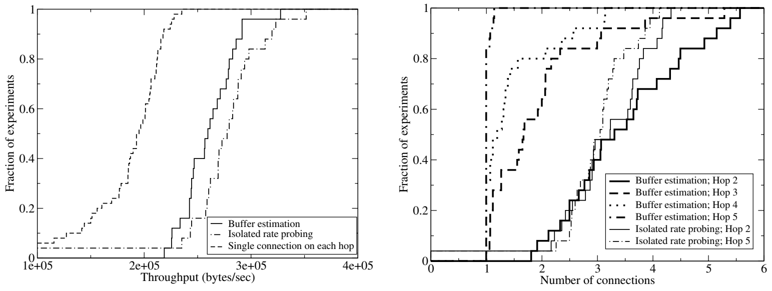
The parameters involved in the implementation and the choice of appropriate values for each parameter, based on our experiments, are discussed below.

Socket Buffer Size: Our experiments with the TCP socket buffer size varying from minimum (8 KB) to maximum (256 KB) showed that the end-to-end throughput of a data transfer over an Overlay-TCP path with a single connection on each hop is higher with the socket buffer size and the receiver advertised window set to the maximum value. This is because the socket buffers can absorb intermittent bursts in either the upstream or downstream connections.

EWMA parameter (γ): γ is the fractional weight of the previous buffer estimate in the exponentially weighted moving average estimator for estimating the current buffer occupancy of the incoming TCP connections. We experimented with values of γ from 0.05 to 0.95, and concluded that a value between the range 0.7 to 0.95 follows the trend well enough and filters out intermediate bursts in the buffer occupancy. In our experiments we used $\gamma = 0.85$.

Estimation and Decision Algorithm Time Interval: In our experiments, we estimate the buffer occupancy every 200 ms. We run the decision algorithm every 30 seconds to determine whether the network conditions have changed enough to either increase or decrease the number of outgoing or incoming connections.

Buffer Occupancy Thresholds (α and β): The buffer occupancy thresholds determine how aggressive the scheme is in using parallel connections. The lower threshold is set to 0.1, below which we consider the socket buffers to be underfull. In order to set the higher threshold, we observed from our experiments on 2-hop paths that if β is too low (0.15), a large number of connections are added and removed on the downstream hop, even when $R_1 < R_2$. On the other hand,



(a) End-to-End Throughput

(b) Number of Connections

Fig. 8. 5-hop overlay path

if β is too high (0.8), the addition of connections is very slow, even when the downstream overlay hop is the bottleneck. In our experiments, we use $\beta = 0.4$.

Maximum connections on each overlay hop (N_{max}): We use $N_{max} = 10$ in our experiments, based on results shown in Section 3.

6.3 Evaluation for a 2-Hop Overlay Path

We performed experiments, as described in Section 6.1, using parameters listed above on a variety of 2-hop overlay paths. On each path, we performed 10-25 experiments. We present representative results for an overlay path with Cornell Univ, Columbia Univ and Stanford Univ as ingress, intermediate and egress nodes respectively. Figure 7(a) shows a CDF of the end-to-end throughput achieved using the proposed schemes and that achieved using a single connection on each hop. The two schemes show similar performance, with the isolated rate probing scheme achieving a slightly higher throughput. Figure 7(b) shows a CDF of average and maximum number of connections started by each scheme. The isolated rate probing scheme is more aggressive, and uses more maximum as well as average connections. The buffer estimation scheme is more conservative and thus achieves lower throughput. We also observed that an increase in degree of mismatch causes a higher factor of improvement in throughput, using either of the two schemes, compared to using a single connection on each hop.

6.4 Evaluation for a Multi-hop Overlay Path

We evaluated the two schemes on multiple overlay paths of 3-5 hops. Figure 8(a) shows a CDF of the throughput achieved using the two schemes, and that achieved using a single TCP connection on each hop, for a sample 5 hop overlay path between Univ of Massachusetts, New York Univ, Georgia Tech, U-Texas, U-Arizona and UCLA. Clearly, the two proposed schemes achieve better throughput than that using a single TCP connection on each hop. The isolated rate probing scheme achieves higher throughput than the buffer estimation scheme.

Figure 8(b) shows a CDF of the average number of connections started by each scheme on different overlay hops. The isolated rate probing scheme starts 2 connections on hops 1, 3, 4, and the buffer estimation scheme starts 1 connection on hop 1. On the bottleneck hop (hop 2), the buffer estimation scheme starts more connections than the isolated rate probing scheme. On other hops, the isolated rate probing scheme is again more aggressive than the buffer estimation scheme.

7 Conclusion

The end-to-end throughput of data transfers in overlay networks that carry data over one or more TCP connections between consecutive overlay nodes is limited by the minimum of the TCP throughputs achievable on each overlay hop. In this work, we aim to maximize the end-to-end throughput of a data transfer on a path in an Overlay-TCP network by using multiple parallel connections on one or more overlay hops. We show that the use of multiple parallel connections on some hops does indeed increase the end-to-end throughput. We propose two schemes that assess the network conditions and dynamically adjust the number of connections used on each overlay hop. We show through experiments on Planetlab nodes that both schemes significantly improve performance while keeping the number of extraneous connections to a minimum. The overlay path design was a first step toward our plan of designing an Overlay-TCP network with the aim of maximizing the throughput of data carried by the entire overlay network. We also plan to study the use of multiple connections on multiple paths between two overlay nodes and its effect on end-to-end throughput. We further aim to study the fair sharing of native link bandwidth between multiple overlay networks.

References

1. Apostolopoulos, J., Wong, T., Wee, S., Tan, D.: On Multiple Description Streaming with Content Delivery Networks. In: IEEE Infocom. (2002)
2. Kwon, G., Byers, J.: Roma: Reliable overlay multicast with loosely coupled tcp connections. In: IEEE Infocom. (2004)
3. Subramanian, L., Stoica, I., Balakrishnan, H., Katz, R.: OverQos: An Overlay based Architecture for Enhancing Internet QoS. In: NSDI. (2004)
4. Padmanabhan, V., Wang, H., Chou, P., Sripanidkulchai, K.: Distributing Streaming Media Content Using Cooperative Networking. In: NOSSDAV. (2002)
5. Keralapura, R., Taft, N., Chuah, C., Iannaccone, G.: Can ISPs take the heat from Overlay Networks? In: HotNets-III. (2004)
6. PlanetLab. (http://www.planet-lab.org/)
7. Banerjee, S., Griffin, T., Pias, M.: The Interdomain Connectivity of Planet lab Nodes. In: Passive and Active Measurement Workshop. (2004)
8. Sundararaj, A., Duchamp, D.: Analytical Characterization of the Throughput of a Split TCP Connection. In: Stevens Inst. Tech Report. (2003)
9. Swany, M., Wolski, R.: Improving throughput with cascaded tcp connections: the logistical session layer. In: UCSB Tech Report. (2002)

10. Lee, B., Balan, R., Jacob, L., Seah, W., Ananda, A.: Avoiding Congestion Collapse on the Internet using TCP Tunnels. Computer Networks (2002)
11. Amir, Y., Danilov, C.: Reliable communication in overlay networks. In: International Conference on Dependable Systems and Networks. (2003)
12. Baccelli, F., Chaintreau, A., Liu, Z., Riabov, A., Sahu, S.: Scalability of reliable group communication using overlays. In: IEEE Infocom. (2004)
13. Urvoy-Keller, G., Biersack, E.: A congestion control model for multicast overlay networks and its performance. In: Networked Group Communication. (2002)
14. Balakrishnan, H., Rahul, H., Seshan, S.: An Integrated Congestion Management Architecture for Internet Hosts. In: ACM SIGCOMM. (1999)
15. Rodriguez, P., Kirpal, A., Biersack, E.: Parallel-Access for Mirror Sites in the Internet. In: IEEE Infocom. (2000)
16. Gkantsidis, C., Ammar, M., Zegura, E.: On the effect of large-scale deployment of parallel downloading. In: IEEE Workshop on Internet Applications. (2003)
17. Peterson, L., Shenker, S., Turner, J.: Overcoming the Internet Impasse through Virtualization. In: HotNets-III. (2004)
18. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation. In: ACM SIGCOMM. (1998)

Randomized Coverage-Preserving Scheduling Schemes for Wireless Sensor Networks[⋆]

Chong Liu, Kui Wu, and Valerie King

Dept. of Computer Science, University of Victoria,
PO Box 3055, STN CSC, Victoria, BC, Canada, V8W 3P6
{chongliu, wkui, val}@cs.uvic.ca

Abstract. Maintaining a long network lifetime with stringent energy constraints on tiny sensor nodes poses an extremely challenging task for wireless sensor networks. This paper provides a thorough analysis on a randomized algorithm that makes scheduling decisions without the help of geographic information. The analytical results precisely describe the relationship among achievable network coverage, energy saving, and node density. We also analyze the performance of the randomized algorithm with time asynchrony and propose a heuristic randomized scheduling scheme to improve the performance.

1 Introduction

Recent progress in wireless communication and MEMS (Micro-ElectroMechanical System) makes it feasible to build tiny wireless sensor nodes that integrate sensors, processors, memory, and wireless transceiver within the size of several cube millimeters [10]. Once deployed, sensor nodes organize themselves into a network through short-range wireless communication. The potential applications of such networks are limitless, ranging from habitant monitoring, battle field surveillance, object tracking, to forest fire alarming.

Due to their extremely small dimension, sensor nodes have very limited energy supply only. In addition, it is usually hard to recharge the battery after deployment, either because the number of sensor nodes is too large, or because the deployment area is hostile for recharging. But once deployed, a sensor network is expected to keep working for several weeks or months. Such expectation will never be met without carefully scheduling the energy consumption of each sensor node to maximize the lifetime of the whole network.

This paper aims at designing and analyzing sensor node scheduling algorithms without geographic information and in the mean time preserving network coverage. We stress that a sensor node's sensing range is totally independent of its radio transmission range because they rely on different hardware. Readers

[⋆] This research was supported by Natural Sciences and Engineering Research Council of Canada.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 956–967, 2005.

should not assume that turning off redundant sensors based on sensing coverage would inevitably result in problems in network connectivity. For instance, we can adopt a broadly-used, two-tiered radio communication architecture, in which sensor nodes communicate directly with a Field Data Collector (FDC) and the FDC communicates directly with the base station.

The contribution of this paper is in four aspects. First, we build a mathematical model to analyze a purely randomized sensor node scheduling algorithm and to illustrate the relationship among achievable coverage quality, energy saving, and node density. Second, we prove that the purely randomized algorithm is resilient to time asynchrony if the network is sufficiently dense. Such feature is indispensable for a practical scheduling algorithm since precise time synchronization is very hard for large sensor networks [2]. Our proof hence provides strong supporting evidence on the randomized scheduling algorithm for realistic applications. Third, we propose a heuristic method that improves the achievable coverage quality of the purely randomized algorithm by evenly assigning neighboring sensor nodes into different monitoring sets. Finally, simulation study is performed to verify the correctness of the analytical results and to demonstrate the advantages of the proposed heuristic method.

2 Purely Randomized Scheduling Scheme

We introduce a purely randomized scheduling algorithm in this section. This algorithm has several prominent features. First, it does not assume the availability of any location or directional information. Second, it is a purely distributed algorithm, thus scalable for large networks. Third, it is resilient to clock asynchrony and requires only a roughly synchronized clock, which significantly decreases the energy and communication overhead required by maintaining network-wide time synchronization. The third feature has never been discovered by previous work [1, 5, 7].

The idea of this algorithm is extremely simple. Assume that the sensor nodes constitute a set S, which will be divided into k disjoint subsets. Each sensor node randomly joins one of the k disjoint subsets. Once the k subsets are formed, they work alternatively. At any given time, there is only one subset working, and all the sensor nodes belonging to this subset will turn on. The intuition behind this algorithm is that when the network is sufficiently dense, each subset alone will cover most part of the field. Because of the randomness, this algorithm cannot guarantee the elimination of blind points– areas that cannot be monitoring by any sensor node for a given time period. But these blind points are not static, that is, a blind point at this time can be covered at another time, as long as it is within the sensing range of certain sensor nodes. This feature makes a phenomenon hidden from detection almost impossible.

The above algorithm has been proposed *simultaneously* in [1, 5] but has not been analyzed thoroughly to obtain practically usable analytical results. In the following sections, we will fill this gap by providing a more straightforward analysis on coverage intensity to evaluate the performance of this purely randomized scheduling scheme.

3 Performance Analysis

3.1 Network Model

We consider static sensor networks in a two-dimensional field. We assume that sensor nodes are randomly and independently deployed in the field. Compared with other sensor deployment strategies such as deployment in grids and deployment according to pre-define positions, random deployment is much easier and cheaper [9]. We assume that all sensors have the same size of the sensing area. Note that our following mathematical model does not make any assumptions on the shape of the sensing area. Therefore, the results obtained from the model are independent of sensor nodes' physical features such as orientation and angular aperture.

3.2 Performance Analysis

Definition 1: Coverage Intensity for a Specific Point. For a given point p in the field, we define the coverage intensity for this point as

$$C_p = \frac{T_c}{T}$$

where T is any given long time period and T_c is the total time during T when point p is covered by at least one active sensor node.

Definition 2: Network Coverage Intensity. We define the network coverage intensity, C_n, as the expectation of C_p. That is, $C_n = E[C_p]$.

For easy reference, all notations used in our probabilistic analysis are listed in Table 1.

Theorem 1: With the purely randomized scheduling algorithm, $C_n = 1 - \left(1 - \frac{q}{k}\right)^n$, where $q = \frac{r}{a}$ is the probability that each sensor node covers a given point.

Proof: Suppose that a given point inside the monitored field is covered by s sensor nodes, denoted as set S. The purely randomized algorithm will assign each sensor node in S to one of the k disjoint subsets randomly. Let's consider the question of how many subsets do not include any sensor node in S. For the first subset (subset 0), it must miss all the s sensor nodes to let the above event happen. Since each sensor node hits the first subset independently with same probability of $\frac{1}{k}$,

$$\Pr\{S_0 \textit{ is empty}\} = \left(1 - \frac{1}{k}\right)^s$$

and thus

$$\Pr\{S_0 \textit{ is not empty}\} = 1 - \left(1 - \frac{1}{k}\right)^s$$

Table 1. Notations

Symbol	Description
n	the total number of deployed sensor nodes
T	the working time duration for each subset in one round
a	the size of the whole field
r	the size of the sensing area of each sensor
k	the number of disjoint subsets
s	the number of sensor nodes that cover a specific point inside the field
S	the set of sensor nodes that cover a specific point inside the field
s_i	the number of sensor nodes that belong to subset i and cover a specific point inside the field
S_i	the set of sensor nodes that belong to subset i and cover a specific point inside the field

This probability is the same for all subsets by symmetry.

We define a random variable X_j. $X_j = 0$ if S_j is empty and $X_j = 1$ otherwise.

Let $X = \sum_{j=0}^{k-1} X_j$ denote the total number of nonempty $S_j, (0 \leq j \leq k-1)$. Then

$$E\left[X\right] = \sum_{j=0}^{k-1} E\left[X_j\right] = k \times \left[1 - \left(1 - \frac{1}{k}\right)^s\right]$$

According to the definition of C_p, the coverage intensity for point p, which is covered by s sensor nodes, is

$$C_p = \frac{E\left[X\right] \times T}{k \times T} = 1 - \left(1 - \frac{1}{k}\right)^s$$

Here s is a binomial random variable, and

$$\Pr\left\{s = j\right\} = \binom{n}{j} \times q^j \times (1-q)^{n-j}$$

where $q = \frac{r}{a}$ is the probability that each sensor node covers a given point.

Therefore, the network coverage intensity C_n, which is the expectation of C_p, can be calculated as

$$C_n = E\left[C_p\right] = 1 - \left(1 - \frac{q}{k}\right)^n$$

□

Corollary 1: For a given k, the lower bound on the number of sensor nodes required in the whole network to provide a network coverage intensity of at least t is

$$\left\lceil \frac{\ln(1-t)}{\ln(1-\frac{q}{k})} \right\rceil$$

where $q = \frac{r}{a}$.

proof: Based on Theorem 1, if we predefine the value of k, which is proportional to the energy saving we target at, and we require the network coverage intensity is no less than a threshold value t, which is the coverage requirement defined by users, we can compute the lower bound of n, the number of sensor nodes required to fulfill the task, by solving the inequality

$$1 - \left(1 - \frac{q}{k}\right)^n \geq t$$

It is easy to see that

$$n \geq \left\lceil \frac{\ln(1-t)}{\ln(1-\frac{q}{k})} \right\rceil .$$

□

Theorem 1 and Corollary 1 illustrate clearly the relationship among the coverage requirement, energy saving, and the minimum number of sensor nodes. Based on Theorem 1, we can also easily get the following corollary:

Corollary 2: For a given n, the upper bound of the number of disjoint subsets to provide a network coverage intensity of at least t is

$$\frac{q}{1 - e^{\frac{\ln(1-t)}{n}}}$$

where $q = \frac{r}{a}$.
Corollary 2 is very useful in dynamically adjusting the coverage intensity of a sensor network after it is deployed. When the total number of sensor nodes is fixed, the network coverage intensity can be adjusted by changing the number of disjoint subsets k. This feature is extremely useful for practical sensor networks requiring adjustable measurement quality and long network lifetime.

4 The Impact of Clock Asynchrony

4.1 A Glance at Clock Asynchrony

Intuitively, the randomized scheduling algorithm should work well without requiring strict time synchronization. Let's check the example shown in Fig. 1.

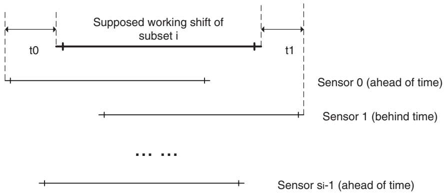

Supposed working shift of subset i
t0
t1
Sensor 0 (ahead of time)
Sensor 1 (behind time)
... ...
Sensor si-1 (ahead of time)

Fig. 1. A point p monitored by s_i sensor nodes in subset i

A point p in the monitored field is covered by s_i sensor nodes in the subset i. Assume that among the s_i sensor nodes, some sensor nodes (e.g., sensor node 0) are ahead of supposed starting time while some (e.g., sensor node 1) are behind the time. In this example, point p can be monitored during the whole working shift of subset i even if the sensor nodes are not synchronized very well.

There are only three possibilities that point p may not be monitored during the working shift of subset i:

1. All the s_i sensor nodes are ahead of the starting time of subset i.
2. All the s_i sensor nodes are behind the starting time of subset i.
3. Some sensor nodes in S_i are ahead of the starting time of subset i while some in S_i are behind the time, and there is a gap period when no sensor node in S_i can monitor point p during the working shift of subset i.

4.2 Analysis on the Impact of Clock Asynchrony

To facilitate analysis, we make the following assumptions:

1. We assume that the internal time ticking frequency of each sensor node is accurate but may not be synchronized precisely to the standard time.
2. We assume that the clock drift of each sensor node from the standard time, Δt, is a normally distributed random variable with parameters $(0, \sigma)$.
3. If we use T, the working duration of each subset in one round, to normalize Δt, we assume $\Delta t \geq \frac{T}{2}$ is an extremely rare case and could be ignored.

For a point p in the field, we suppose there are s_i sensor nodes assigned to subset i $(1 \leq i \leq k)$ covering p. Let Δt_j denote the deviation of the clock of the j-th sensor node from the standard clock $(0 \leq j \leq s_i - 1)$. Δt_j is a normally distributed random variable with parameters $(0, \sigma)$. If $\Delta t_j \leq 0$ holds for all j $(0 \leq j \leq s_i - 1)$, which means all the clocks of these s_i sensor nodes are ahead of time, there will be a period of unmonitored time at the end of the working duration of subset i with the length of $\min\{-\Delta t_j, 0 \leq j \leq s_i - 1\}$. Likewise, if $\Delta t_j \geq 0$ holds for all j $(0 \leq j \leq s_i - 1)$, which means all the clocks of these s_i sensor nodes are behind time, there will be a period of unmonitored time at the beginning of working duration of subset i with the length of $\min\{\Delta t_j, 0 \leq j \leq s_i - 1\}$.

Note that the sensor nodes with an ahead-of-time clock in subset $i + 1$ and the sensor nodes with a behind-time clock in subset $i - 1$ could help decrease the unmonitored time length during the working duration of subset i. Nevertheless, considering these cases will greatly increase the analysis complexity by introducing correlation between neighboring subsets, we ignore these cases when calculating the network coverage intensity. Therefore, the calculated network coverage intensity is the lower bound of the actual value.

We now calculate the expectation of the unmonitored time fraction (the time when p is not covered by any of these s_i sensor nodes) during the working shift of subset i. We denote this expectation as E_{s_i}.

When $s_i = 0$, it is obvious that $E_0 = 1$. When $s_i \geq 0$,

$$E_{s_i} = \int_0^{\infty} x f_1(x)\, dx + \int_{-\infty}^{0} -y f_2(y)\, dy$$

where $x = \min\{\Delta t_j, 0 \leq j \leq s_i - 1\}$, $y = \max\{\Delta t_j, 0 \leq j \leq s_i - 1\}$, $f_1(x)$ and $f_2(y)$ are the *p.d.f.* of x and y, respectively.

Since $\Delta t_0, \Delta t_1, \cdots, \Delta t_{s_i-1}$ are independent normal distribution random variables, we can get

$$Pr\{x \geq a\} = [1 - \Phi(a)]^{s_i}$$

where $\Phi(a)$ is the *c.d.f.* of normal distribution. Therefore,

$$f_1(x) = s_i \phi(x) [1 - \Phi(x)]^{s_i - 1}$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}$ and $\Phi(x) = \int_{-\infty}^{x} \phi(x)\, dx$.
By symmetry, we have

$$\int_{-\infty}^{0} -y f_2(y)\, dy = \int_0^{\infty} x f_1(x)\, dx$$

Therefore,

$$E_{s_i} = 2 \int_0^{\infty} s_i x [1 - \Phi(x)]^{s_i - 1} \phi(x)\, dx.$$

Since x is a normal distribution with parameters $(0, \sigma)$, when $x \geq 0$, we have $\Phi(x) \geq \frac{1}{2}$, and thus $1 - \Phi(x) \leq \frac{1}{2}$.
So we get

$$E_{s_i} \leq 2 \int_0^{\infty} s_i x \left(\frac{1}{2}\right)^{s_i - 1} \phi(x)\, dx = \frac{s_i \sigma}{\sqrt{2\pi}} \left(\frac{1}{2}\right)^{s_i - 2}$$

Here, E_{s_i} is the expectation of the unmonitored time fraction during the working shift of subset i, which includes exactly s_i sensor nodes covering the point p. Suppose that the total number of sensor nodes in the network that cover point p is s. Any subset may contain j sensor nodes to cover p, where j varies from 0 to s. Therefore, we can calculate the expectation of the unmonitored time fraction for any subset (denoted as $\bar{E}_s$):

$$\begin{aligned}
\bar{E}_s &= \sum_{j=0}^{s} E_j \times Pr\{the\ subset\ contains\ j\ nodes\ to\ cover\ p\} \\
&\leq 1 \times \left(1 - \frac{1}{k}\right)^s + \sum_{j=1}^{s} \frac{j\sigma}{\sqrt{2\pi}} \left(\frac{1}{2}\right)^{j-2} \left(1 - \frac{1}{k}\right)^{s-j} \left(\frac{1}{k}\right)^j \binom{n}{j} \\
&= \left(1 - \frac{1}{k}\right)^s + \frac{s\sigma}{8\sqrt{2\pi}(k-1)} \left(1 - \frac{1}{2k}\right)^{s-1}
\end{aligned}$$

Thus, for any point covered by s sensor nodes, the expectation of the monitored time fraction of the working-shift of any subset

$$E_s = 1 - \bar{E}_s \geq 1 - \left(1 - \frac{1}{k}\right)^s - \frac{s\sigma}{8\sqrt{2\pi}(k-1)} \left(1 - \frac{1}{2k}\right)^{s-1}$$

We next calculate E, the expectation of E_s.

$$\begin{aligned}
E &= \sum_{s=0}^{n} E_s \times Pr\{the\ point\ is\ covered\ by\ s\ sensor\ nodes\} \\
&= \sum_{s=0}^{n} E_s \binom{n}{s} q^s (1-q)^{n-s},\ where\ q = \frac{r}{a} \\
&\geq 1 - \sum_{s=0}^{n} \left(1 - \frac{1}{k}\right)^s \binom{n}{s} q^s (1-q)^{n-s} \\
&- \frac{s\sigma}{8\sqrt{2\pi}\,(k-1)} \sum_{s=0}^{n} \left(1 - \frac{1}{2k}\right)^{s-1} \binom{n}{s} q^s (1-q)^{n-s} \\
&= 1 - \left(1 - \frac{q}{k}\right)^n - \frac{nq\sigma}{8\sqrt{2\pi}\,(k-1)\,(1-q)} \left(1 - \frac{q}{2k}\right)^{n-1}
\end{aligned}$$

For any point p, by symmetry, each subset has the same E value, so the expectation of the monitored time fraction, which is the network coverage intensity C_n, according to the definition, can be calculated as

$$C_n = \frac{k \times E}{k} = E$$

Observing the expression of C_n above, we find that the term $1 - \left(1 - \frac{q}{k}\right)^n$ is equal to the C_n in Section 3.2, where all the clocks are well-synchronized. Thus, the last term $\Delta = \frac{nq\sigma}{8\sqrt{2\pi}(k-1)(1-q)} \left(1 - \frac{q}{2k}\right)^{n-1}$ indicates the impact of time asynchrony on network coverage intensity. The numeric results in Fig. 2 show the weight of Δ over C_n, when $\sigma = 0.1613$. In this case, $Pr(\Delta t \geq \frac{T}{2})$ is less than 0.1% and thus negligible. The small values of the weight indicate the negligible impact of clock asynchrony on network coverage and illustrate that the purely randomized scheduling scheme is resilient to time asynchrony.

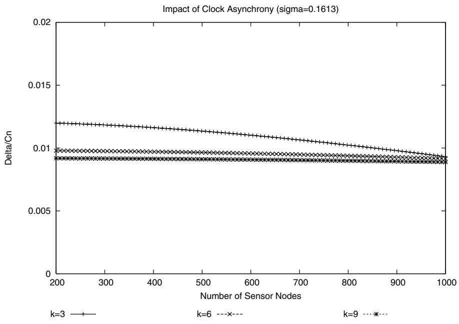


Fig. 2. Impact of Clock Asynchrony

5 Heuristic Randomized Scheduling Scheme

The purely randomized scheduling scheme works well, but it still has room for further improvement. For instance, assume that a point p is covered by s ($s \geq k$) sensor nodes. If the s sensor nodes are assigned to only l subsets, where $l < k$, the point p is only covered in $\frac{l}{k}$ time fraction even if the achievable coverage intensity of the point p is 100%. Our heuristic randomized scheduling algorithm is based on the above observation and tries to assign the s sensor nodes evenly into the k subsets.

To achieve even assignment of sensor nodes to the k subsets, we adopt the following strategy. Initially, each sensor node selects a random backoff time between 0 and *maxBackOff*. The value of *maxBackoff* should be reasonably large to avoid broadcast collision. During the backoff period, each sensor node will listen to the channel and receive broadcast messages from its neighbors that indicate their subset selections. At the timeout of its backoff, each sensor node checks the decisions on subset assignment received from its neighbors, assigns itself into a subset that includes the fewest neighbors (if a tie exists among several subsets, select one randomly among these subsets), and broadcasts this decision to its neighbors. Once all sensor nodes have selected their subset, they will work in the same way as in the purely randomized scheduling algorithm.

Unlike the distributed greedy algorithms proposed in [1] and [8] which depend on the availability of precise location information of each sensor node to decide redundancy, our heuristic method is simple and does not assume the availability of any location information. The energy cost of our method is only *one local* broadcast for each sensor node, which is extremely lightweight and scalable for large sensor networks.

6 Simulation

6.1 Simulation Model

In our simulation, all sensor nodes are deployed randomly in a 1000 meters × 1000 meters square area. Each sensor node has a fixed sensing range of 50 meters. We set the radio transmission range equal to the sensing range when we compare the performance of the purely randomized scheduling algorithm and the heuristic scheduling algorithm, since the performance of the former one has nothing to do with radio range while the latter one performs best with the sensing range equal to the radio range. The performance of the heuristic scheduling scheme under different ratios of the sensing range over the radio range is also investigated.

We use network coverage intensity (refer to Section 3.2) to evaluate the performance of the scheduling algorithms. For each simulation scenario, twenty runs with different random seeds are conducted and the results are averaged.

6.2 Simulation Results

Fig. 3 illustrates the relationship between the number of deployed sensor nodes and network coverage intensity with the purely randomized scheduling scheme.

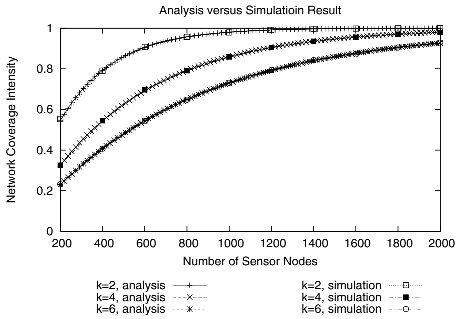


Fig. 3. Analytical and Simulation Results on Purely Randomized Scheduling

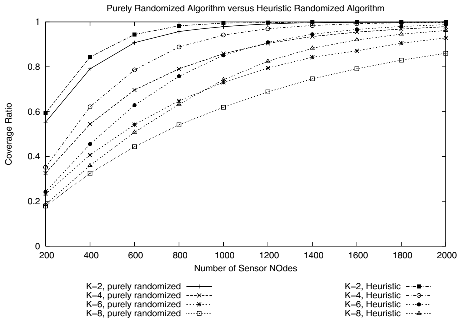


Fig. 4. Comparison of Purely and Heuristic Randomized Scheduling Schemes

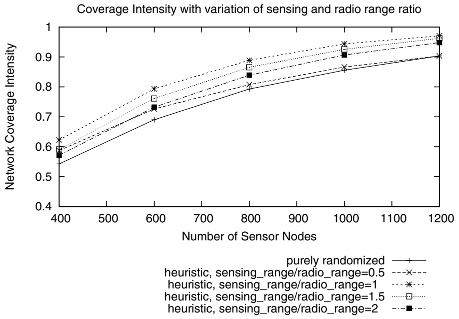


Fig. 5. Coverage Intensity with Different Ratios of Sensing Range over Radio Range

Both analytical results and simulation results are presented in the figure. From the figure, the analytical results and simulation results match pretty well, indicating the correctness of our mathematical analysis. Given a fixed k, network coverage intensity increases with the increase of the number of deployed sensor nodes, and given a fixed number of deployed sensor nodes, network coverage intensity increases with the decrease of k. It is consistent with the intuition since decreasing k or increasing the number of sensor nodes will increase the density of working nodes and hence improves the coverage intensity. This figure also provides a reference map between network coverage intensity and the number of sensor nodes needed.

Fig. 4 shows the performance comparison between purely randomized scheduling and heuristic randomized scheduling. From the figure, it can be seen that given the same k value and the same number of sensor nodes, the heuristic scheme always achieves a higher coverage intensity than the purely randomized one. In addition, if there are enough sensor nodes deployed, the heuristic scheme can outperform purely randomized scheme even if the purely randomized scheme use a smaller k. This can be verified by the fact that the two curves corresponding to the "k=6, purely randomized" scheme and the "k=8, heuristic" scheme cross when the number of deployed sensor nodes is around 900. This phenomenon further demonstrates the advantage of the heuristic scheme.

Fig. 5 shows the impact of different radio ranges on the performance of heuristic randomized scheduling. The best performance appears in the case where the sensing range and the radio range are equal. This is because under this situation, the number of neighbors (in terms of radio range) of a sensor node accurately reflects the local coverage redundancy in the neighborhood of this sensor node. If the sensing range is different from the radio range, the estimation of sensing redundancy will be inaccurate, misleading the heuristic method to make wrong decisions and thus degrading its performance. From Fig. 5, if the ratio of the sensing range over the radio range is not too far from 1, the performance of the heuristic randomized scheduling scheme is still better than the purely randomized scheduling scheme. But the advantage of heuristic method will be negligible when the radio range differs significantly from the sensing range and provides very inaccurate redundancy information.

7 Conclusion

In this paper, we analyze the performance of a purely randomized scheduling algorithm and disclose the relationship among coverage intensity, energy saving, and the required number of sensor nodes. Our mathematical model is significantly different from other work [1] in that our model uses a straightforward performance metric and provides simple equations to estimate performance directly from given parameters. This analysis is very important in the sense that it greatly facilitates dynamical adjustment of network coverage intensity. We also analyze the performance of the randomized algorithm with time asynchrony and propose a heuristic randomized scheduling scheme to improve the performance.

References

1. Z. Abrams, A. Goel, and S. Plotkin, "Set k-cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks ," *Proceedings of Information Processing in Sensor Networks 2004*, Berkeley, CA, April 2004.

2. J. Elson, and K. Romer, "Wireless Sensor Networks: A New Regime for Time Synchronization," *Proceedings of First Workshop on Hot Topics in Networks*, Princeton, NJ, October 2002.
3. W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, Third Edition, John Wiley & Sons, 1968.
4. C. Hsin, and M. Liu, "Network Coverage Using Low Duty-Cycled Sensors: Random & Coordinated Sleep Algorithm," *Proceedings of Information Processing in Sensor Networks 2004*, Berkeley, CA, April 2004.
5. C. Liu, "Randomized Scheduling Algorithm for Wireless Sensor Networks," *Course Project of CSc 523 (Randomized Algorithms)*, University of Victoria, March 2004.
6. S. Mequerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks," *Proceedings of IEEE INFOCOM 2001*, Anchorage, AL, April 2001.
7. S. Slijepcevic and M. Potkonjak "Power Efficient Organization of Wireless Sensor Networks," *Proceedings of IEEE International Conference on Communications 2001*, Helsinki, Finland, June 2001.
8. D. Tian, and N.D. Georganas, "A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks," *Proceedings of ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, October 2002.
9. S. Tilak, N.B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure tradeoffs for sensor networks," *Proceedings of First International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, USA, September 2002.
10. B. Warneke, M. Last, B. Leibowitz, K.S.J. Pister, "Smart Dust: Communicating with a Cubic-Millimeter Computer," *IEEE Computer Magazine*, Vol.34, No.1, January 2001, pp. 44-51.
11. K. Whitehouse and D. Culler, "Calibration as Parameter Estimation in Sensor Networks", *Proceedings of First International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, USA, September 2002.
12. F. Ye, G. Zhong, J. Cheng, S. Lu and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks," *Proceedings of the 10th IEEE International Conference on Network Protocols* , Paris, France, November 2002.

Dynamic Scheduling for Scalable Media Transmission over cdma2000 1xEV-DO Broadcast and Multicast Networks

Kyungtae Kang[1], Jinsung Cho[2,⋆], Yongwoo Cho[1], and Heonshik Shin[1]

[1] School of Computer Science and Engineering,
Seoul National University, Seoul, 151-744, Korea
{ktkang, xtg05, shinhs}@cslab.snu.ac.kr
[2] Dept. of Computer Engineering,
Kyung Hee University, Youngin, 449-701, Korea
chojs@khu.ac.kr

Abstract. The cdma2000 1xEV-DO mobile communication system provides broadcast and multicast services (BCMCS) to meet an increasing demand for multimedia data services. Currently, broadcast and multicast streams are scheduled using a slot-based static algorithm, which fails to support dynamic environments where broadcast contents are added or removed on-line. We propose a dynamic packet-scheduling algorithm that works with a retransmission scheme to enable scalable and adaptive service across cdma2000 1xEV-DO BCMCS environments. Integrated with earliest deadline first (EDF) real-time scheduling, the proposed algorithm not only adapts efficiently to dynamic contexts but also satisfies the real-time requirements of broadcast streams. Furthermore, by exploiting the fine granular scalable (FGS) characteristics of the MPEG-4 Part 2 standard, our scheme can adapt to a changing resource environment more flexibly. Simulation results show a significant improvement in average playback quality.

Keywords: cdma2000 1xEV-DO BCMCS, earliest deadline first (EDF), resource allocation, fine granular scalable (FGS).

1 Introduction

Recently, work has begun, in both the Third Generation Partnership Project (3GPP) and in the 3GPP2 group, on enhancing 3G networks to support multimedia broadcast and multicast services. The 3GPP2 group recently baselined the specification for a cdma2000 high-rate broadcast packet-data air interface (BCMCS[1]) [1][2]. Their goal is to design a system that can deliver multimedia broadcast and multicast traffic while minimizing resource usage by both

[⋆] Corresponding author.

[1] In this paper, the term 'BCMCS' stands for cdma2000 1xEV-DO BCMCS.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 968–979, 2005.

the radio access and core networks. In addition, users expect minimum latency when joining or leaving the network, and multimedia streams must be delivered continuously as mobile users move around. A hierarchical design with localized multicasting and local servers is necessary to provide a scalable system.

In this paper, we propose a dynamic scheduling algorithm combined with a retransmission scheme for the cdma2000 1xEV-DO BCMCS scheduler. The current static scheduling algorithm which is part of the BCMCS specification cannot adapt to an environment where content streams change dynamically. When a video stream is newly registered by mobile nodes, empty slots must be found to service this changed context. As the transmission rates of video streams differ from each other, the slot periods of these streams will also differ, which causes the serving slots to overlap. Also, if there are no slots left, the existing static scheduler is unable to service the new stream because it is impossible to adjust the bit-rate in a static scheme. Our proposed scheduling algorithm deals with such dynamic situations by exploiting the inherent attributes of the earliest deadline first (EDF) [3] algorithm. Our algorithm also works on the assumption that the video streams are fine granular scalable (FGS). By dynamically adjusting the scalability of the video streams that are already being serviced, new video streams can be admitted, although the playback quality is somewhat degraded. However the average playback quality degradation of all mobile nodes can be reduced compared to the current static scheme. Three policies to adjust the quality of each video stream are proposed, and their performance is compared experimentally. We also show how to control the admission of packets with a utilization bound test, using the EDF scheduling algorithm.

BCMCS currently employ Reed-Solomon coding to correct errors at the cost of air resources. If the channel condition is good, many slots will be overbooked due to the space taken by parity information, which reduces the number of empty slots available for a newly registered video stream. By adding a retransmission scheme to the EDF scheduling algorithm, we can save many slots when the condition of a channel is good, and guarantee the minimum playback quality by protecting the base-layer packets in FGS coding.

The remainder of this paper is organized as follows: in Section 2, we introduce our research background, and Section 3 gives the motivation for our proposed scheduling algorithm followed by a description. Simulation results are discussed in Section 4 and we conclude in Section 5.

2 Background

2.1 Packet Scheduling in BCMCS

In a unicast environment, 1xEV-DO employs a time-shared forward link that serves users sequentially in a time-multiplexed manner. The fundamental timing unit for forward link transmissions is a 1.67 ms slot that contains the pilot and MAC channels, and a data portion that may contain traffic or a control channel. The 1xEV-DO system adopts a dynamic data rate transmission scheme which allows efficient data transmission with a data rate that varies according to the channel condition of each mobile node [4].

In a broadcast or multicast environment, however, a fixed transmission rate must be used, because it is impossible to reflect the link state of all mobile nodes. BCMCS video streams are delivered to one or more nodes in 1.67 ms time-slots that statically in advance the current static scheduler allocates. In receiving a cdma2000 1xEV-DO broadcast or multicast video stream, a mobile node obtains the content from the video server in pre-determined slots, but the service parameters are transmitted as an overhead message. When a mobile node requests a BCMCS video stream, packets are transmitted from the Packet Data Serving Node (PDSN) to the Packet Control Function (PCF), which uses a time-stamp to make sure that the packet is received simultaneously at all base stations. The mobile node maintains its soft state by registering periodically. If no mobile node is left registered to it, the video stream is eventually terminated [5].

For error recovery, the broadcast MAC protocol does not use an ARQ-based error control scheme, because the same content will be delivered to many users simultaneously. Instead of handling retransmission requests individually, BCMCS use a Reed-Solomon forward error-correcting code. A Reed-Solomon encoder takes a block of digital data and adds extra ‘check’ bits. The encoded data is larger and therefore uses more physical slots, which means using more air resource. As a result, fewer video streams can be serviced than would be the case without Reed-Solomon coding.

2.2 Scalable Media Stream

There has been a lot of extensive research on scalable video transport and resource management over mobile wireless networks, focusing on unicast services [6][7][8]. The fine granularity scalable (FGS) video coding scheme [9] in MPEG-4 [10] encodes video frames into multiple layers, including a base layer of video of relatively lower quality and several enhancement layers containing increasingly more detailed data that are used to enhance the base layer and hence to improve the quality of the final video. The progressive fine granularity scalable (PFGS) coding scheme [11] represents an advance on the FGS scheme.

These scalable video coding schemes feature a layered bit-stream structure in which different layers have different levels of importance. The base layer stream is very sensitive to channel errors. If the decoder finds any errors in the lower layers, the higher layers of the current frame are discarded whether they are correct or not. If they go undetected, errors in the base layer will propagate to the start of the next group of pictures (GOP) and cause serious drifting problems in the higher layers of the following frames. While accuracy in the base layer is essential for decoding video streams, the enhancement layers are more tolerant of channel errors. Errors in the enhancement layers corrupt video quality only in the current frame and a few subsequent frames, and the deterioration is not as noticeable as that caused by errors in the base layer. Because of this layered bit-stream structure, it seems reasonable that the more important layers should be provided with more protection against errors than the less important layers. But the error recovery scheme in the current BCMCS standard makes no allowance for the different value of the different layers, and all are equally likely to be damaged during transmission. Our adaptive framework confronts this problem, and has the following key features:

- **Graceful quality degradation:** unlike non-scalable video, scalable video can adapt its representation to bandwidth variations. When the network has to drop packets, it does so with awareness of the video that the data represents. As a result, perceptual quality is gracefully degraded even under severe channel conditions.
- **Efficiency:** when there is excess bandwidth, it is used efficiently in a way that maximizes the perceptual quality or the revenue.
- **Fairness:** the resource can be shared to enhance the average playback quality. We propose three policies to share slot resources efficiently.
- **Guaranteed delivery of the base layer:** the base layer stream is guaranteed by giving it more opportunities for retransmission. This avoids abrupt drops in service quality.

3 The Proposed Scheme

From an established service scenario [5], we can infer that current static scheduling algorithms cannot adapt to an environment in which broadcast flows start and stop in part of the system, affecting the resources available. For example, when a video stream is added, empty slots must be found to schedule this new content. Because the transmission rates of video streams differ from each other, the slot periods of these video streams will also differ. Thus, the slots used by different streams may overlap, which can cause problems in servicing them. Additionally, we can infer that, if all slots are allocated for predetermined video streams, additional video streams or newly created asynchronous video streams (newsflashes or other irregular content) that are not predetermined would never be serviced. This inefficient slot utilization is exacerbated by the parity data required for Reed-Solomon coding. In addition, the current scheduling scheme makes it impossible to change the quality of a video stream in accordance with bandwidth variations. This motivates us to deploy a dynamic scheduling algorithm. Specifically, we propose to apply the EDF real-time scheduling algorithm, which endeavors to deliver packets before their deadlines, together with retransmission to maximize the slot utilization in BCMCS.

When many video servers provide multimedia streams, each BCMCS flow is continually split into packets which then wait to be serviced. If a mobile node cannot receive a packet, that packet is reported as lost or damaged to the base station, via the reverse unicast channel. Because successful receipt of a broadcast packet will give satisfaction to more users than receipt of a retransmitted packet, the scheduler gives a higher priority to broadcast packets than to retransmitted packets, and base-layer packets are given higher priority than enhancement-layer packets. Thus, broadcast packets will be transmitted earlier, giving them more chance of retransmission in case of loss. Then, if any slots are still available, base-layer packets are transmitted, and finally enhancement-layer packets.

Table 1. Parameters used in this paper

Parameter	Description
b_i	Bit rate of each BCMCS IP flow (kbps)
p_i	Period of τ_i in units of slots
R_i	Buffer requirement of each BCMCS IP flow
D_i	Relative deadline slot for the periodic multicast video frame τ_i
D_{ak}	Deadline slot for retransmitted packets $\bar{\tau}_k$
F_i	Size of transmitted data during one period for each video stream
e_i	Number of slots required to forward video stream during one period
$\bar{e}_r$	Number of slots required to retransmit lost packets of $\bar{\tau}_r$
M_i	Number of bytes that can be forwarded in one slot (data payload)
U_P	Utilization of periodic video frames
U_{cur}	Current slot utilization
τ_i	Periodic broadcast video frame
$\bar{\tau}_k$	Aperiodic retransmitted video frame (packets)
A_k	Arrival time of $\bar{\tau}_k$
N	Total number of mobile nodes in a cell
N_{τ_i}	Number of mobile nodes registered for video flow τ_i

Admission control and resource allocation are tightly related to the packet scheduling algorithm. In our scheme, the admission of newly created video frames and retransmitted packets is allowed if sufficient resources for the adequate delivery of data to a mobile node are available. In the EDF scheduling algorithm, resource availability is checked by a utilization bound test. In our scheme, admission is also controlled by a utilization bound test, as follows: Suppose there are n BCMCS flows currently being serviced in the cell and the bit-rate of each BCMCS flow is b_i kbps. Then the scheduler must periodically transmit b_i kbits per second, since BCMCS flow corresponds to a periodic task in a real-time system. Thus, each flow has the period p_i. If the buffer requirement (R_i) to guarantee continuous playback is $K * p_i$, then the relative deadline slot is $(K * p_i)/\varphi$ where φ is the fundamental timing unit of a slot and has the value of 1.67 ms. Using the transmission scheme in cdma2000 1xEV-DO, e_i is determined by $\lfloor b_i/M_i \rfloor$. Thus the admission of a new multicast video stream (τ_j) is decided using the following equation [3]:

$$U_P = \sum_{i=0}^{n-1} \frac{e_i}{Min(p_i, D_i)} + \frac{e_j}{p_j} \leq 1 \ . \qquad (1)$$

If $U_P > 1$, the scheduler adjust the scalability of existing video streams to permit the new video frame. We consider three policies to adjust the scalability of each video stream, experimentally comparing their performance with respect to the peak signal-to-noise ratio (PSNR) value:

- **Policy 1 (Fair degradation):** the appearance of a new video stream produces equal quality degradation in all existing flows.

- **Policy 2 (Victim degradation):** victim flows are selected for adjustment to permit the newly requested video flow.
- **Policy 3 (Proportionally fair degradation):** the degradation in quality of each existing flow is inversely proportional to the number of subscribers that it has.

There may also be packets for retransmission, generating aperiodic tasks which increase slot utilization in proportion to their number. If the current slot utilization is U_{cur}, then the average slot utilization by retransmitted packets for one mobile node will be $(U_{cur} - U_P)/N$. When a request to service new video flow is made, the bit-rates of existing flows have to be adjusted. A reserve capacity $(U_{reserved})$ of $(U_{cur} - U_P) * (1 + N_{\tau_j}/N)$ must be provided for retransmitted packets, which leaves $1 - U_{reserved}$ to be allocated to periodic broadcast flows, by application of the policies just discussed.

Using the first policy, when m new video flows are requested, and the number of available slots is insufficient, the bit-rate of each flow i is degraded in an egalitarian manner. The degraded bit-rate is given as follows:

$$b_i^{'} = \left\lfloor b_i * \frac{1 - U_{reserved}}{U_{cur} + \sum_{j=0}^{m} \frac{e_j}{p_j}} \right\rfloor . \tag{2}$$

Using the second policy, victims are selected for a degraded flow. If the bit-rate of the first flow drops below the base-layer bit-rate, then another victim is selected from among those flows which are not already victims: we choose the one with the smallest number of subscribers. The bit-rates of additional victim flows are degraded in a similar manner to the first. Using the third policy, the bit-rate is degraded in proportion to the number of subscribers. If the bit-rate of the flow with the lowest subscriber count falls below the base-layer bit-rate, then the bit-rate of that flow is fixed at the minimum value, and the other flows are scaled down appropriately. The adjusted bit rate $b_i^{'}$ of video flow i is given as follows:

$$Degradation = \left(\sum_{i=0}^{n-1} b_i + \sum_{n=0}^{j=m-1} b_j)(1 - \frac{1 - U_{reserved}}{U_{cur} + \sum_{j=0}^{m} \frac{e_j}{p_j}} \right) \tag{3}$$

$$b_i^{'} = Degradation * w_i , \tag{4}$$

where w_i is a weight which is inversely proportional to the mobile node count of video flow i, selected from among the values $\frac{N_{\tau_i}}{N}$, $\frac{N_{\tau_j}}{N}$ $(0 \leq i \leq n-1,$ $0 \leq j \leq m-1)$.

Admission of newly generated retransmission packets can be decided in the following way [12][13][14]: Suppose the aperiodic task set $Q(t) = \{\bar{\tau}_0, \bar{\tau}_1, \bar{\tau}_2, ...\bar{\tau}_k, ...\}$ and the periodic task set $P = \{\tau_0, \tau_1, \tau_2, ...\tau_{i-1}\}$ are to be scheduled by the EDF algorithm. For $\bar{\tau}_k \in Q(t)$, $\bar{e}_k(t)$ is defined as the remaining execution time of $\bar{\tau}_k$ at time t, and we define the set S to be the union of $Q(t)$ and P. For all $\bar{\tau}_k \in Q(t)$, $S(t, \infty)$ is schedulable if and only if $\forall k, U_{S(t,\infty)}(\bar{\tau}_k) \leq 1$, where

$$U_{S(t,\infty)}(\bar{\tau}_k) = \frac{\sum_{i=0}^{n-1} e_i(t) + \sum_{\bar{\tau}_r \in Q((t,\infty),HP(D_{ak}))} \bar{e}_r(t)}{D_{ak} - t} . \quad (5)$$

The term $HP(D_{ak})$ represents 'higher priority'. If $\bar{\tau}_k$ is included in the base-layer packets, then $\bar{\tau}_j$ are tasks which is included in the base-layer packets and have an earlier deadline than D_{ak}. If $\bar{\tau}_k$ is included in the enhancement-layer packets, then all retransmission packets corresponding to the base layer or the enhancement layer, with deadlines earlier than D_{ak}, are included in $HP(D_{ak})$. Thus, base-layer packets have a higher priority than enhancement-layer packets. If two packets correspond to the same layer, the packet with the earlier deadline has the higher priority. Using Equation (5), we might now expect to be able to determine the feasibility of a new retransmission request. However this test is actually impossible because an infinite number of tasks $(Q((t,\infty),HP(D_{ak})))$ would have to be considered. So we use the following recursive method to calculate an upper bound on the utilization.

A task set $S(t,\infty)$, $Q(t) = \{\bar{\tau}_1, \bar{\tau}_2, \bar{\tau}_3, ..., \bar{\tau}_{k-1}, \bar{\tau}_k, ...\}$ is sorted by priority. Thus, $\bar{\tau}_{k-1}$ has a higher priority than $\bar{\tau}_k$, and an aperiodic task $\bar{\tau}_k \in Q(t)$ is schedulable when $\bar{U}(\bar{\tau}_k) \leq 1$. The virtual finish time f_k is defined as the sum of the execution times (number of slots required) of all periodic tasks and of the aperiodic tasks that have a higher priority than $\bar{\tau}_k$ during the time period $[A_k, D_{ak}]$. Thus

$$U(\bar{\tau}_k) \leq \frac{max\{f_{k-1} - A_k, 0\} + \bar{e}_k + U_P * (D_{ak} - D_{a(k-1)})}{D_{ak} - A_k} = \bar{U}(\bar{\tau}_k) \quad (6)$$

$$f_{k-1} = A_{k-1} + U(\bar{\tau}_{k-1}) * d_{k-1} \; . \quad (7)$$

Using these equations while fixing U_k and F_k, the schedule can be analyzed within constant time. Hence, we can ignore the admission control overhead.

4 Performance Evaluation

4.1 Simulation Model

Fig. 1 shows our experimental architecture for broadcasting scalable video from a base station to mobile nodes. Admission control and resource reservation for retransmitted packets are two of the major factors in controlling the transmission rate, as explained in the previous section. If a request is accepted, rate-controlled video packets are scheduled in an EDF manner.

We conducted experiments using the Foreman testbench video sequences streamed at 30 frames per second with a total of ten thousand frames. Each video stream is handled with our reference MPEG-4 FGS codec, which is derived from the framework of the European ACTS Project Mobile Multimedia Systems (MoMuSys) [15] and modified for our purpose. It consists of a 120 kbps base-layer bit-rate and a 100 kbps enhancement-layer bit-rate with 120 levels

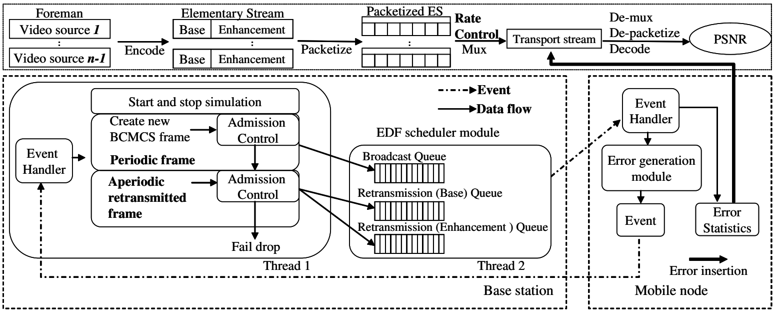


Fig. 1. Simulation architecture

(1 kbps unit). All video streams are sent through channel coding and packetized before being transferred via a cdma2000 1xEV-DO physical slot. In the experiment, we used QPSK modulation with a 1228.8 kbps data-rate forward channel.

At the mobile node, successful receipt of the video packet is determined by an error generation module, using the simple threshold model suggested by Zorzi [16]. When an error occurs, this module may subsequently request retransmission of packets by means of a retransmission event generated by the event generation module. The affected packets are then inserted into a queue by an event handler in the base station, where they wait to be serviced by the scheduler. In our simulation environment, the air channel is simulated by a two-state Markov model, which can simulate the error sequences generated by data transmission channels. These errors occur in clusters or bursts with relatively long error-free intervals between them. In the following equations, $1-\alpha$ (or β) is the probability that the i-th packet block is successfully transmitted, given that the (i-1)-th block was successful (or unsuccessful). Fading in the air channel is assumed to have a Rayleigh distribution. The steady-state error rate ε is then obtained as follows:

$$\varepsilon = \frac{\alpha}{\alpha + \beta} \ . \tag{8}$$

If the Rayleigh fading margin is F, the steady-state error rate can be given as

$$\varepsilon = 1 - e^{-\frac{1}{F}} \ . \tag{9}$$

We can now derive the values of α and β from the following equations, and from Equations (8) and (9). F is the *fading margin* and the parameter $f_d N_{BL} T$ (the Doppler frequency normalized to the data rate where N_{BL} represent the block (packet) length) is taken as 0.02. *Burst* is the average length of packet errors and is given by $1/\beta$, where

$$\beta = \frac{Q(\theta, \rho\theta) - Q(\rho\theta, \theta)}{e^{\frac{1}{F}} - 1}, \text{ and } \theta = \sqrt{\frac{2/F}{1-\rho^2}} \ . \tag{10}$$

The term ρ is the correlation coefficient of two samples of the complex Gaussian fading process, and is expressed as $\rho = J_0(2\pi f_d T)$. Additionally,

$$Q(x, y) = \int_{y}^{\infty} e^{-\frac{x^2+\kappa^2}{2}} I_0(x\kappa) w \; d\kappa \tag{11}$$

is the Marcum-Q function. Thus the relationship between the steady-state error rate and the Markov parameter can be represented as

$$\beta = \frac{1-\varepsilon}{\varepsilon}\left[Q(\theta, \rho\theta) - Q(\rho\theta, \theta)\right] \text{ , where} \tag{12}$$

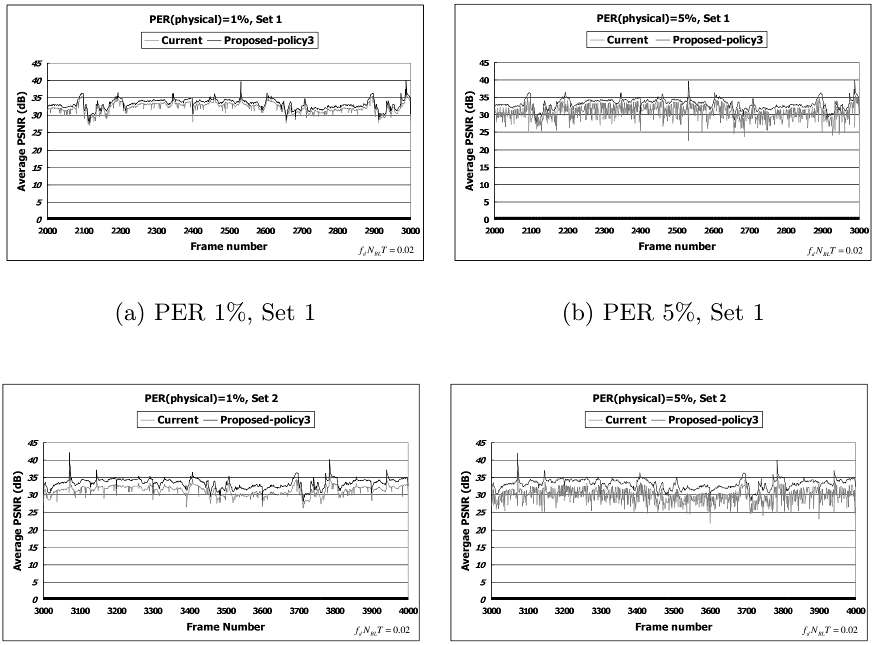
$$\theta = \sqrt{\frac{-2\log(1-\varepsilon)}{1 - J_0^2(2\pi f_d N_{BL} T)}} \; . \tag{13}$$

We now compare our retransmission model with the current Reed-Solomon-based BCMCS error correction scheme. The code (16, 12, 4) is used in our experiments. To evaluate two error recovery schemes, errors are injected into the original transport stream and the PSNR of the resulting video stream is calculated to estimate the quality of a reconstructed image compared with an original image. Finally, we compare the current Reed-Solomon-based static scheduling scheme with the proposed retransmission-based dynamic scheduling scheme with respect to PSNR.

4.2 Experimental Result

In our experiment, we used two sets of data: Set 1 is $\{(\tau_i, N_{\tau_i}) | (\tau_0, 5), (\tau_1, 2), (\tau_3, 5), (\tau_4, 12), (\tau_5, 20)\}$, ($\tau_1$ is a new video flow), Set 2 is $\{(\tau_i, N_{\tau_i}) | (\tau_0, 5), (\tau_1, 2), (\tau_2, 2), (\tau_3, 5), (\tau_4, 12), (\tau_5, 20)\}$ and τ_1 and τ_2 are new video flows. U_{BCMCS} is the slot utilization when frames are encoded by Reed-Solomon coding, and $U_{Proposed}$ is the slot utilization when the proposed retransmission scheme is used. Before any new video flows are created , $U_{BCMCS} = 1$, which means that no available slot is left. But $U_{Proposed} = 0.75$, because 25% of the slots are saved in comparison to the current scheme, which corresponds to the space that would have been used by parity data. These saved slots are used to recover lost packets by retransmission.

Fig. 2 shows the average PSNR of all mobile nodes using data Sets 1 and 2 when the physical-layer packet error-rates ($PER_{physical}$) are 1% and 5%. We sampled from Frame 2000 to Frame 3000 of Set 1, and from Frame 3000 to 4000 of Set 2. From these graphs, we can see that average PSNR improves when a new video flow is admitted to a cell and bandwidth is shared with the current existing video flows. We contrast this with the situation in the current scheme, where the new flow would never be admitted. The gap between the two curves (current scheme vs. proposed scheme) in the graph increases as more flows share the bandwidth. The large fluctuations in the lower curve come from the failure of base-layer packets, an effect which is exacerbated as the physical-layer packet error-rate ($PER_{physical}$) increases. When a base-layer packet is broken, the video quality is abruptly degraded.



(a) PER 1%, Set 1

(b) PER 5%, Set 1

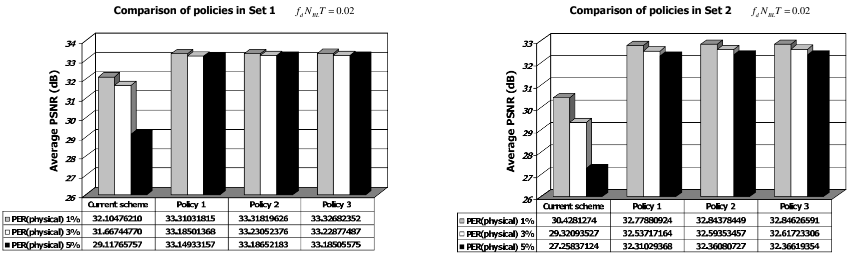
(c) PER 1%, Set 2

(d) PER 5%, Set 2

Fig. 2. Average PSNR of all mobile nodes for Set 1 and Set 2

In Fig. 3(a) and 3(b), three proposed policies are compared with each other and with the current scheme using the experimental data Sets 1 and 2. As a new video flow is admitted, the average PSNR per flow is somewhat degraded. But we can see that the overall average is much improved compared to the current scheme. When the $PER_{physical}$ is 5% and Policy 3 is used, the overall average improves as much as 18.7% compared to the current scheme, as shown in Fig. 3(b). We can improve the average playback quality by considering the distribution of mobile nodes which subscribe to each video flows although it is a relatively small effect compared to quality adjustment and packet retransmission. Because the dispersion of subscriber numbers in our experimental set is large, the third policy produces the best quality. The average PSNR using Policy 3 for Set 2 is approximately 0.05 dB higher than that using Policy 1. Using Set 1, however, when the $PER_{physical}$ is 3% or 5%, Policy 2 gives a slightly better result. The final result will also depend to an extent on the dispersion of subscriber numbers and on the shape of the curve relating bit-rate to PSNR, an observation which can be used to optimize the resource allocation to maximize the average playback quality.

Fig. 4 compares the average PSNR of nodes for each video flow when the $PER_{physical}$ is 3%. As a new flows are admitted, the bit-rate of each video flow is adjusted according to the three proposed policies. In the current scheme, new flows (Flow 1, 2) can never be serviced because existing flows already occupy



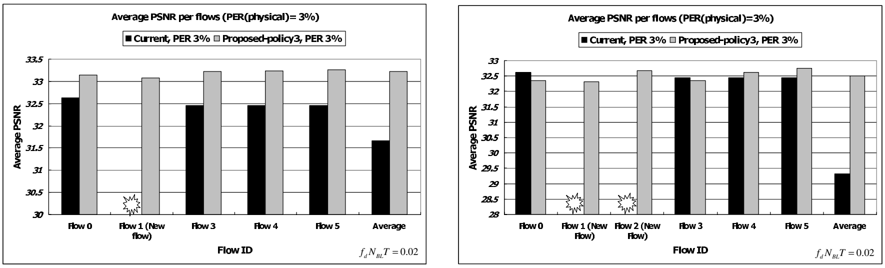
	Current scheme	Policy 1	Policy 2	Policy 3
▨ PER(physical) 1%	32.10476210	33.31031815	33.31819626	33.32682352
□ PER(physical) 3%	31.66744770	33.18501368	33.23052376	33.22877487
■ PER(physical) 5%	29.11765757	33.14933157	33.18652183	33.18505575

(a) Set 1

	Current scheme	Policy 1	Policy 2	Policy 3
▨ PER(physical) 1%	30.4281274	32.77880924	32.84378449	32.84626591
□ PER(physical) 3%	29.32093527	32.53717164	32.59353457	32.61723306
■ PER(physical) 5%	27.25837124	32.31029368	32.36080727	32.36619354

(b) Set 2

Fig. 3. Comparison of policies



(a) Set 1

(b) Set 2

Fig. 4. Average of PSNR with respect to flows

all the slots, which are allocated statically. Thus the average PSNR of each new video flow in Fig. 4 is zero, which degrades the average PSNR. This degradation grows as the number of subscribers to new video flows increases.

5 Conclusion

We have proposed a dynamic scheduling algorithm based on EDF to handle the situation where broadcast channels change dynamically. Our algorithm can adapt to a dynamic context where the bandwidth requirement of all video flows fluctuate, and it also meets the real-time requirement of broadcast streams. In addition, we prevent abrupt quality degradation of broadcast streams by deploying a retransmission scheme, instead of the Reed-Solomon coding used in the current BCMCS, to cope with poor channel condition. By dynamically adjusting the scalability of the video streams that are already being serviced, even when no available slots remain, new video streams can be admitted, although the playback quality is somewhat degraded. Extensive simulation has shown the efficiency of our scheme compared with that used by the current BCMCS. By protecting the base layer by means of retransmission, we can avoid abrupt drops in service quality. Our experimental results show that the average PSNR of all mobile nodes in our proposed scheme is higher than in the current BCMCS scheme. In addition, by considering the distribution of subscribers across the video flows, we can further improve the average playback quality.

References

1. P. Agashe, R. Rezaiifar, P. Bender and QUALCOMM, "Cdma2000 high rate broadcast packet data air interface design," *IEEE Communications Magazine*, vol. 42, no. 2, pp. 83-89, February 2004.
2. 3GPP2 C.S0024 v3.0, "Cdma2000 high rate packet data air interface specification," December 2001.
3. F. Cottet, J. Delacroix, C. Kaiser and Z. Mammeri, *Scheduling in Real-Time Systems*, Wiley, pp. 31-32, October 2002.
4. G. D. Mandyam and Y. C. Tseng, "Packet scheduling in CDMA systems based on power control feedback," *Proc. IEEE International Conference on Communications*, vol. 9, pp. 2877-2881.
5. J. Wang, R. Sinnarajaj, T. Chen, Y. Wei, E. Tiedemann and QUALCOMM, "Broadcast and multicast services in cdma2000," *IEEE Communications Magazine*, vol.42, no. 2, pp. 76-82, February 2004.
6. D. Wu, T. Hou and Y.-Q. Zhang, "Scalable video transport over wireless IP networks," *Proc. 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication*, vol. 2, pp. 1185-1191, September 2000.
7. C.-H. Yeh, "Scalable, adaptive, and reliable resource management in highspeed and mobile networks," *Proc. 10th Conference on Computer Communications and Networks*, pp. 182-189, October 2001.
8. K. Gao, W. Gao, S. He, P. Gao and Y. Zhang, "Real-time scheduling on scalable media stream delivery", *Proc. International Symposium on Circuits and Systems*, vol. 2, pp. 25-28, May 2003.
9. W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301-317, March 2001.
10. ISO/IEC 14496-2, "Coding of audio-visual objects - part2: visual," May, 2004.
11. F. Wu, S. Li and Y. Q. Zhang, "A framework for efficient progressive fine granularity scalable video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 332-344, March 2001.
12. J. Park, M. Ryu and S. Hong, "Utilization demand analysis: An integrated admission control aproach for a mixed set of periodic and aperiodic tasks," *Proc. Korea Information Science Society Spring Conferences*, pp. 391-393, April 2004.
13. M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," *Real-Time Systems*, vol. 10, no. 2, pp. 179-210, March 1996.
14. B. K. Choi, D. Xuan, R. Bettati, W. Zhao and C. Li, "Utilization-based admission control for scalable real-time communication," *Real-Time Systems*, vol. 24, no. 2, pp. 171-202, March 2003.
15. A. Pearmain, A. Carvalho, A. Hamosfakidis and J. Cosmas, "The MoMuSys MPEG-4 mobile multimedia terminal," *Proc. 3rd ACTS Mobile Summit Conference*, pp. 224-229, June 1998.
16. M. Zorzi, R. R. Rao and L. B. Milstein, "Error statistics in data transmission over fading channels," *IEEE Transactions on Communication Communications*, vol. 46, no. 11, pp. 1468-1477, November 1998.

Proxy-Assisted Scheduling for Energy-Efficient Multimedia Streaming over Wireless LAN[*]

Fan Zhang[**] and Samuel T. Chanson

Department of Computer Science,
Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong

Abstract. Energy is a critical resource for battery-driven mobile and wireless devices. The power-saving mode (PSM) provided by the 802.11 standard is not adequate for mobile clients running streaming media applications. We propose new power-aware scheduling strategies that enhance existing traffic shaping schemes for the purpose of minimizing the communication energy consumption of mobile clients. Scheduling decisions are made by the local proxy and executed at the access point. We consider two cases. In the first case, the proxy is unaware of the power characteristics of the clients. We present an optimal scheduling scheme that minimizes the time that mobile clients stay in high-power modes. This scheme achieves significant energy savings compared to using traffic shaping alone. In the second case, the power profiles and the residual battery capacities of the clients are taken into account in generating the schedule. Unlike most existing work, we believe these factors should be considered in the performance metric. We propose a dynamic programming approach that computes the optimal transmission schedule based on the new metric. We also present an efficient heuristic which exhibits near-optimal performance in the simulation experiments.

1 Introduction

Battery-driven mobile/wireless devices such as PDAs and smartphones need to be designed to utilize energy efficiently. As more and more mobile devices are network-aware instead of being used in the stand-alone mode, this work aims to minimize the energy consumption of the wireless network interface, which has become one of the major energy consumers in mobile systems. For handheld devices such as PDAs, communication power can be as high as 40% of the total system's power [1].

A Wireless LAN (WLAN) card typically has three high-power modes: transmit, receive and idle. Data transmission consumes the highest level of power. An idle interface consumes almost the same amount of power as when it is receiving data. Furthermore, most wireless interfaces also have a low-power sleep/doze mode which consumes

[*] The work described in this paper was supported by grants from the National Natural Science Foundation of China and the Research Grants Council of Hong Kong SAR, China (Project No. N_HKUST618/02 and HKUST6178/03E).

[**] Corresponding author. Email:zhangfan@cs.ust.hk, Tel: (+852)6026-9658, Fax: (+852)2358-1477.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 980–991, 2005.

significantly less power. For example, a typical IEEE 802.11b WLAN card consumes 1400mW, 950mW and 805mW while transmitting, receiving and being idle, respectively. While sleeping, it consumes only 60mW [2].

Data communication for common mobile users in a WLAN is asymmetric, and is dominated by downlink traffic. Most of the time, a client is receiving data over the wireless link, say, watching online videos or browsing the web. The uplink traffic primarily consists of short control messages such as HTTP requests and TCP ACK packets.

Multimedia streaming applications over wireless networks are gaining popularity as wireless bandwidth increases. The IEEE 802.11b standard offers a transmission rate of 11Mbps, while the 802.11a and 802.11g can achieve up to 54Mbps. The bandwidth growth enables users to listen to music, view video clips, and even watch movies online on their mobile devices equipped with wireless interfaces. Since these applications usually run over a long period of time and involve a large volume of data, a substantial amount of energy is consumed for receiving the media streams over the wireless link.

The current Power-Saving Mode (PSM) supported by the IEEE 802.11 standards is not adequate for media streaming applications. According to the specification, all mobile clients wake up periodically and listen to the beacon frames broadcast by the Access Point (AP) at the beginning of each beacon interval (typically 100ms) [3]. If the beacon indicates data pending, then the wireless interface must stay at the high-power mode and listen to the channel until all data are received. The waiting time is not specified by the standards and can be arbitrarily long. It is obvious the energy dissipation while waiting is wasteful and should be minimized. Furthermore, existing approaches do not consider the power characteristics and the residual battery capacities of the mobile clients when scheduling streaming media traffic from the AP. If a client at a low battery level is kept waiting for a long time, its battery will be quickly depleted.

In this work, we propose power-aware traffic scheduling schemes to address the above issues. As all traffic must go through the local proxy first, in our scheme, scheduling decisions are made at the proxy and delivered to the AP for execution (see Figure 1). Instead of transmitting media frames individually, the proxy groups consecutive frames and sends them in a burst. More importantly, knowing the communication bandwidth and the workload, the proxy determines the optimal transmission schedule that minimizes the waiting times of all the clients. As the time of receiving data is fixed and cannot be changed, the total energy consumption is minimized by minimizing the sum of the waiting times. In addition, the proxy may communicate with the mobile clients and obtain useful information for scheduling such as residual battery capacity and the idle power consumption rate of their wireless interfaces. With the information, we use a weighted energy consumption function as a new performance metric, and propose an optimal scheduling algorithm that minimizes this metric.

Although proxy/server-assisted traffic shaping has been studied before, the energy saving capability of traffic shaping alone is limited without an appropriate scheduling algorithm. The power-aware scheduling algorithms we propose in this work enhance the traffic shaping technique and achieve additional energy savings. A detailed comparison with some existing schemes is presented in Section 5.

The rest of the paper is organized as follows. The system architecture and a motivating example are introduced in Section 2. We present our proxy-assisted scheduling schemes in Section 3. In Section 4, the performance of the proposed schemes is evaluated and compared with some existing schemes. Following a review of related work in Section 5, Section 6 concludes this paper.

2 Background

2.1 System Architecture

The WLAN environment considered is illustrated in Figure 1. Heterogeneous wireless clients are associated with an AP and access the Internet via the proxy server. The 802.11b standard, which is the most prevalent setting in today's wireless deployment, is used between the mobile clients and the AP. The AP is typically connected to a proxy server by a high-speed local area network (e.g., 100Mbps or 1Gbps Ethernet). We assume the communication delay between the proxy and the AP is negligible.

Mobile clients retrieve continuous media streams from remote web servers over the Internet. Each client plays back only one stream at a time. The applications (such as stored video and audio) are assumed to be able to tolerate a relatively long startup delay. However, media frames should be delivered to the clients according to the specified frame rate to ensure smooth playback.

All the data streams go through the proxy, so the proxy server at the application layer can understand the semantics of the streams, including data rate and frame rate. The function of the proxy is two-fold. First, it receives data packets from the Internet and stores them in local buffers. Second, the proxy delivers the packets to the AP at an appropriate time so they can be sent to the mobile clients via the wireless link. As streaming media applications usually utilize the UDP protocol, uplink ACK messages are generally not required.

2.2 802.11 Power-Saving Mode

The current 802.11b standard supports a mechanism called the Power-Saving Mode (PSM) to reduce the wireless energy dissipation of mobile clients. The PSM allows a client to switch to the low-power sleep mode when it has no outgoing or incoming traffic. The client needs to notify the AP that it is going to enter the sleep mode. The AP then buffers all the data destined to that client when it is sleeping.

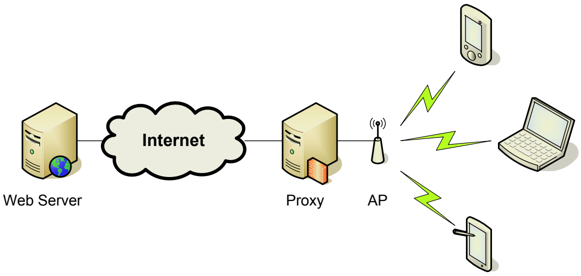


Fig. 1. System architecture

All sleeping mobile clients periodically wake up and listen to the beacon broadcast at periodic *beacon intervals*. The number of beacon intervals between consecutive wake-ups is called the *listen interval*. The beacon includes a Traffic Indication Map (TIM) that informs the mobile clients of data pending. If a client has data pending, it sends a *Power-Saving poll (PS-poll)* message to the AP to indicate it is awake. The client must then actively monitor the channel until all its data are received. A client that does not have data pending can switch back to the low-power sleep mode immediately. The time and energy overhead due to transition between the active mode and the sleep mode is very short and is usually considered to be negligible.

Upon receiving a PS poll message, the AP could send data packets immediately to the client or reply with a short ACK message and delay data transmission in order to receive PS-poll messages from the other clients. The second scheme, called *deferred response*, allows the AP to determine the transmission sequence adaptively. Our proposed schemes make use of this feature, where the AP carries out the transmission schedule produced by the proxy.

3 Proxy-Assisted Scheduling

We propose two power-aware scheduling schemes to accompany the traffic shaping mechanism. The transmission schedule is computed at the proxy server. We assume the AP can communicate with the proxy, receive the transmission schedule and send out data packets accordingly. The AP notifies a client with the TIM only if the data of that client is scheduled to be sent in the current beacon interval. This function can be implemented in the form of application level software or firmware in the AP. Throughout this work, we do not consider transmission errors or packet retransmission, and assume the time for transmitting control packets such as RTS/CTS and PS-poll to be negligible.

3.1 Energy-Efficient Scheduling

We first consider the case where the power characteristics the clients are not known for scheduling. In this case, the total energy is minimized by minimizing the total time that the clients stay in the active (receive/idle) modes.

Suppose there are n streams destined to n receiving clients, respectively. The proxy groups consecutive data frames in each stream and sends them in a single burst. Each burst contains a number of media frames that can sustain continuous playback for B time units. Therefore each client only needs to receive data once every B time units. We call B the *buffer length*. Suppose the length of the beacon interval is L, then $B = m\,L$ where m is a positive integer. We use t_i $(1 \leq i \leq n)$ to denote the time required to transmit each data burst of stream i. A feasible schedule is an assignment of the n bursts $t_1, t_2, \ldots, t_n$ into m consecutive beacon intervals $I_1, I_2, \ldots, I_m$ each of length L.

The scheduling problem is formulated as follows. We use $S_j = \{s_{j,1}, \ldots, s_{j,n_j}\}$ to denote the order of transmission in the interval I_j, where $s_{j,k}$ indicates the data stream for the k-th transmission. The objective function is the total active time of all the clients and can be expressed as:

$$\sum_{j=1}^{m}\sum_{k=1}^{n_j}(n_j - k + 1) \cdot t_{s_{j,k}}. \tag{1}$$

The above problem is equivalent to scheduling independent tasks on identical machines to minimize the sum of their finish times (i.e., completion times). The latter problem can be reduced to a min-cost-flow problem and can be solved efficiently [4].

```
Input: $n, m, t_i$ $(1 \le i \le n)$.
Output: $S_j$ $(1 \le j \le m)$.

Sort streams in non-increasing order of $t_i$ $(1 \le i \le n)$;
for $i = 1, 2, \ldots, n$
    $j = i \bmod m$;
    Add $i$ to $S_j$;
for $j = 1, 2, \ldots, m$
    Sort $S_j$ in non-decreasing order of $t_{j,k}$;
```

Fig. 2. Algorithm for calculating optimal schedule

We present a polynomial time algorithm to build the optimal schedule [5]. First the streams are sorted by the transmission time in non-increasing order. Then stream i is assigned to interval I_j where $j = i \bmod m$. Finally, in each beacon interval, streams are transmitted in non-decreasing order of the transmission time. The algorithm is formally presented in Figure 2, and the time complexity is $O(n \log n)$.

For example, suppose $m = 3$ and there are 6 streams whose receiving times are $10, 9, 8, 7, 6, 5$, respectively. According to the algorithm, stream i is allocated to interval $(i \bmod m)$. As a result, streams $\{1, 4\}$ are allocated to interval 1, streams $\{2, 5\}$ to interval 2, and streams $\{3, 6\}$ to interval 3. In each interval, the allocated streams are scheduled in non-decreasing order of their receiving times. Therefore stream 4 is scheduled before stream 1 in interval 1, and so on. The total time in high-power modes of all the clients is 63.

The condition given in Theorem 1 below guarantees the schedule produced by the above algorithm is feasible. Please refer to [5] for the proof of optimality.

Theorem 1. *Suppose t_i $(1 \le i \le n)$ are sorted in non-increasing order. If the inequality $(\sum_{i=1}^{n} t_i)/m + t_1 \le L$ holds, then the schedule produced by the algorithm in Figure 2 is feasible, that is, $\forall j$, $\sum_{k=1}^{n_j} t_{s_{j,k}} \le L$.*

Proof: We prove the theorem by contradiction.

Suppose $(\sum_{i=1}^{n} t_i)/m + t_1 \le L$ holds but $\exists j$, $\sum_{k=1}^{n_j} t_{s_{j,k}} > L$. Now the last stream transmitted in interval I_j is s_{j,n_j}. Obviously $t_{s_{j,n_j}} \le t_1$. As the schedule is optimal in minimizing the sum of finish times, moving stream s_{j,n_j} to any other interval cannot decrease the sum of finish times. Therefore, we have

$$\sum_{k=1}^{n_l} t_{s_{l,k}} \ge \sum_{k=1}^{n_j - 1} t_{s_{j,k}} > L - t_{s_{j,n_j}} \ge L - t_1, \quad \forall l \ne j,\ l = 1, 2, \ldots, n.$$

Furthermore,

$$\sum_{i=1}^{n} t_i = \sum_{j=1}^{m} \sum_{k=1}^{n_j - 1} t_{s_{j,k}} > m \cdot (L - t_1),$$

which contradicts with $(\sum_{i=1}^{n} t_i)/m + t_1 \leq L$. □

Given today's high wireless bandwidth and that many handheld devices support low-data-rate media streams only, the above feasible condition usually holds. In this case, the algorithm given in Figure 2 produces the optimal solution for our scheduling problem. However, if the condition does not hold, the schedule produced may not be feasible. Then the problem of determining whether all the streams can be feasibly scheduled is NP-hard [6].

In the next section, we present a dynamic programming approach to minimize the weighted sum of finish times. The algorithm has a pseudo-polynomial time complexity. The problem of minimizing the sum of finish times, as a special case, can also be solved using this dynamic programming approach.

3.2 Battery-Aware Scheduling

The mobile devices in a WLAN environment are likely to be produced by several different manufacturers, and their wireless interfaces may have different power characteristics. For example, the idle power of an OriNOCO PC Gold card is 805mW, while that of a Cisco AIR-PCM350 card is more than 30% higher (1080mW) [2]. In addition to power dissipation, the available battery capacities of the clients may also be different. The battery capacity is approximately proportional to the battery size. Limited by size, small mobile devices such as smart phones have lower battery capacities compared to PDAs or Pocket PCs.

The transmission schedule calculated in Section 3.1, based on minimizing the finish times, may not be optimal when the power profiles of the clients are heterogeneous. Furthermore, without considering the residual energy capacity, the schedule may require a client at a low battery level to wait for a long time and quickly deplete its battery.

Based on the above observations, we propose to use a weighted energy consumption function as the performance metric. The idea is to minimize the waiting time of the mobile clients with a high idle power and a low battery level. Denote the idle power of client i as P_i and the residual battery capacity as C_i. The weight $w_i = P_i/C_i$ is the proportion of the residual battery that will be consumed by keeping the wireless interface idle for one time unit.

Assume n_j clients receive their data in interval j, the problem is formally formulated as a linear optimization problem with 0/1 variables $x_{i,j,k}$ (see Figure 3). The binary variable $x_{i,j,k} = 1$ indicates that client i is the k-th to receive data in interval j. The first constraint specifies that the total transmission time in each interval cannot exceed the length of the interval L. The second constraint guarantees every stream is transmitted and only transmitted once. The last constraint ensures only one stream is selected in each interval and each transmission sequence in the schedule is generated.

Solving this optimization problem is NP-hard [4], so we employ a dynamic programming approach. First the streams are sorted by t_i/w_i in non-decreasing order. The solution procedure takes n stages. At stage i, the minimum weighted energy for

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{n_j}\Big(w_i\, x_{i,j,k}\cdot\sum_{l=1}^{k}\sum_{i=1}^{n} t_i\, x_{i,j,l}\Big)\\
\text{subject to} \quad & \sum_{k=1}^{n_j}\sum_{i=1}^{n} t_i\, x_{i,j,l} \le L, \quad j=1,\dots,m\\
& \sum_{j=1}^{m}\sum_{k=1}^{n_j} x_{i,j,k} = 1, \quad i=1,\dots,n\\
& \sum_{i=1}^{n} x_{i,j,k} \le 1, \quad j=1,\dots,m; k=1,\dots,n_j\\
& x_{i,j,k}\in\{0,1\}, \quad \forall k, 1\le k\le n.
\end{aligned}$$

Fig. 3. A linear optimization formulation

transmitting the data streams to clients $1,\dots,i$ is determined. Given a schedule, let $T_j\ (j=1,\dots,m)$ be the latest finish time of the last transmission in interval j. We use $E_i\,[T_1,\dots,T_m]$ to denote the minimum weighted energy for transmitting data streams $1,\dots,i$ so that the finish times in the m intervals are $T_1,\dots,T_m$, respectively. The weighted energy of the optimal schedule for n clients is therefore $E_n\,[L,\dots,L]$.

The energy function in the initial stage is:

$$E_0\,[T_1,\dots,T_m] = \begin{cases} 0 & \text{if } \forall j,\, T_j = 0,\\ \infty & \text{otherwise.}\end{cases} \tag{2}$$

The optimal weighted energy is obtained by iteratively computing E_1 up to E_n. The energy function in each stage i is given by:

$$E_i\,[T_1,\dots,T_m] = \min_{1\le j\le m}\Big\{E_{i-1}\,[T_1,\dots,T_j - t_i,\dots,T_m] + w_i\, t_i\Big\}, \quad 1\le i\le n. \tag{3}$$

This dynamic programming solution has a pseudo-polynomial time complexity. For any given m-tuple $[T_1,\dots,T_m]$, the time complexity of computing the optimal $E_i\,[T_1,\dots,T_m]$ according to equation (3) is $O(m)$. As there are at most L^m different states for each stage i, the time complexity for all n stages is $O(m\,n\,L^m)$. If the data streams are not sorted, the time complexity becomes $O(m\,n\,L^m + n\,\log n)$. The complexity is exponential to the number of beacon intervals, so the computation time is prohibitive for large m. Therefore, we present an efficient heuristic below which has polynomial-time complexity.

We modify the optimal scheduling algorithm for minimizing the sum of finish times in Section 3.1 and use it under our new performance metric. The heuristic takes three steps to compute the transmission schedule. First, it sorts the streams in non-increasing order of $t_i/w_i\ (1\le i\le n)$. Then it schedules data stream i to interval $j = i \bmod m$. Finally, the algorithm sorts the streams scheduled in each interval in non-decreasing order of $t_i/w_i\ (1\le i\le n)$. The time complexity of the heuristic is $O(n\,\log n)$. In the next section, we show that this heuristic exhibits near-optimal performance.

4 Performance Evaluation

Two sets of experiments have been performed to evaluate the schemes proposed in Sections 3.1 and 3.2. We have built a simulator to simulate traffic shaping and scheduling in a 11Mbps wireless LAN environment. In the simulated environment, 20 clients are simultaneously running streaming media applications. The media streams have constant data rates which are uniformly distributed between 64bps and 450bps, representing applications from music streaming to video streaming. We set the beacon interval at 100ms. Each client wakes up periodically and listens to each beacon to see if there is pending traffic. The idle/receiving power of the wireless interface is uniformly distributed between 500mW and 1500mW with a supply voltage of 5V.

4.1 Energy Minimization

In the first set of experiments, we evaluated the sum of the energy consumption rates of all the clients in the idle and receiving modes, comparing the proposed scheduling scheme in Section 3.1 with a scheme that employs traffic shaping only. In the latter scheme, streams are assigned to the intervals in a round-robin manner. The buffer length B was varied from 100ms to 500ms (i.e., 1 to 5 beacon intervals). For each buffer length, 10 experiments were conducted in which the data rates of the streams were varied. The average of the total energy consumption rates in the 10 experiments was taken and presented in Figure 4. The x-axis denotes the buffer length. The y-axis specifies, under the given traffic shaping and scheduling schemes, the energy consumed by all the clients in one second.

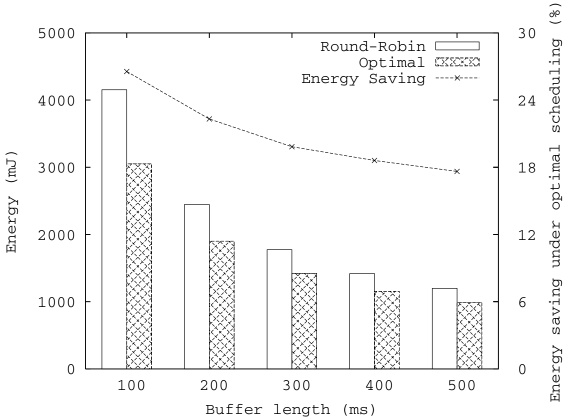


Fig. 4. Total energy consumption rate with traffic shaping and scheduling

A few observations can be made regarding the results. First, traffic shaping with optimal scheduling achieves about 18% to 27% energy savings compared to plain traffic shaping with round-robin scheduling. Second, the energy consumption rate under both schemes decreases as the length of stream buffering increases. With a longer buffering time, the clients need to wait for data reception in fewer beacon intervals, which reduces the energy consumption in the waiting mode. Third, as the buffer length increases, the difference between optimal scheduling and round-robin also decreases. This is because with fewer clients receiving data in each beacon interval, it is less flexible and beneficial to change the data transmission sequence.

4.2 Battery-Aware Performance

In the second set of experiments, we compared the performance of four algorithms in minimizing the weighted sum of finish times. In addition to round-robin, we implemented the optimal dynamic programming approach and the heuristic presented in Section 3.2. The optimal algorithm in Section 3.1 essentially minimizes the sum of finish times of all the clients disregarding their power dissipation and battery capacities. This algorithm is also evaluated under the new performance metric. Due to the exponential computation time of the dynamic programming approach, in the experiments we used only three values of the buffer length: 100ms, 200ms, and 300ms. As in the previous set of experiments, for each buffer length setting, 10 experiments were conducted and the average value was taken.

We set the maximum residual battery capacity to 5000mAh and the minimum to 100mAh. The battery capacities of the other clients were uniformly distributed between these two bounds.

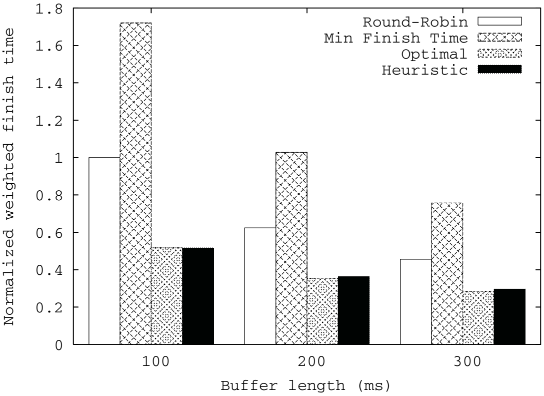


Fig. 5. Normalized weighted sum of finish times with traffic shaping and scheduling

The simulation experiment results are presented in Figure 5. We set the baseline to the weighted sum of finish times under round-robin scheduling with a buffer length of 100ms. All results are normalized with respect to the baseline for easy comparison. Note that the algorithm that minimizes the sum of finish times performed even worse than round-robin under the new metric. The performance of the heuristic algorithm is close to that of the optimal algorithm (less than 5% difference) in all cases, and is about 50% better than that of the round-robin algorithm.

After evaluating the algorithms under the system-wide new metric, we took a closer look at the energy dissipation of each individual client. In particular, we compared the

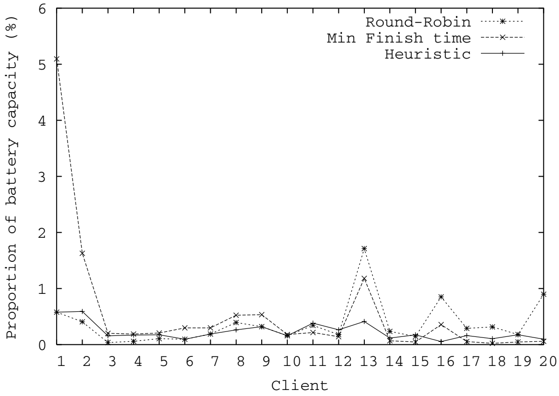


Fig. 6. Proportion of residual battery capacity to receive 1-minute of media stream

proportion of the residual battery capacity each client spent in receiving one minute of media stream. As shown in Figure 6, we observed that the optimal algorithm for minimizing finish times created very unbalanced energy dissipation among the clients: some clients used only 0.02% while others used as much as 5% of their battery capacity.

Some clients spending a large portion of their residual energy, in particular clients 1, 13, 16 and 20, were found running at a low battery level and requesting media streams with a high data rate. The min-finish-time algorithm, disregarding their battery information, scheduled some of these clients to wait and receive data towards the end of each beacon interval. Therefore, these clients will deplete their battery power sooner than the other clients. In contrast, our proposed heuristic produced the most balanced schedule: all clients consumed less than 0.6% of their remaining battery capacities for data reception per minute. Furthermore, 70% of the clients consumed between 0.1% and 0.3%. The variance of the energy consumption among different clients was small. The battery-aware mechanism is particularly beneficial to those clients running at a low battery level. For example, client 13 consumed 1.7% and 1.2% of its residual battery capacity for receiving one minute of data under the min-finish-time and the round-robin schemes, respectively , but it consumed only 0.4% under the battery-aware heuristic.

Powered with battery-awareness, our proposed scheduling scheme effectively minimizes the waiting times of clients at a low battery level, which in turn reduces their energy consumption and prolongs their battery life.

5 Related Work

As energy conservation has become an important design issue in mobile computing systems, there has been a lot of work on energy-efficient communications [7,8,9,10].

Much work has focused on minimizing energy consumption during data transmission [7,8]. Based on the Shannon equation of channel capacity, the transmission rate can be traded off for a lower transmission power. When the workload is fixed, an optimal transmission schedule can be computed to minimize the total transmission energy [7,8].

On the other hand, when energy and time are limited, a transmission schedule can be built to maximize the data throughput or the value from data transmission [10].

Dynamic mode transition complements transmission power control by switching the wireless interface from high-power active modes to a low-power mode whenever possible. In [11], the energy consumption for accessing a web page is analyzed. Based on information such as the expected size of the page, the network round trip time (RTT) and the mode transition overhead, the mobile client can determine whether the wireless interface should be switched off before the requested page is received. Krashinsky and Balakrishnan [9] have investigated the interaction between TCP and the IEEE 802.11 power-saving mechanism. They suggest that the mobile client does not need to listen to every beacon. Instead, the client can elect to wake up adaptively based on traffic density. On the other hand, a rendezvous policy [12] requires the proxy broadcasts a transmission schedule while the clients turn on their wireless interfaces at the scheduled times. Utilizing an additional low-power transceiver at the client is studied in [2]. The low-power transceiver is always monitoring the channel. If it receives a control message indicating data pending, it wakes up the main transceiver to start receiving data.These schemes do not utilize traffic shaping technique at the proxy and require modification to the clients so they can understand the traffic characteristics and switch modes accordingly. Our approach, instead, utilizes the PSM mechanism of the IEEE 802.11 standard, which is usually already built into the system software of the clients.

Traffic shaping increases data burstiness and reduces the time intervals the wireless interface must stay at the high-power transmit/receive modes [13,14]. Chandra [13] has studied the energy savings of server-side traffic shaping on different streaming media formats. For client-server style multimedia applications, a control channel can be established to facilitate power management [14]. The client informs the server of its expected service rate and instantaneous buffer occupancy. The server then adjusts the amount of data to be delivered in each burst. However, as the server is usually located at a distant site over the Internet, energy savings may be adversely affected due to obsolete client information and distorted data bursts. Note that none of the above schemes make use of any optimized scheduling algorithm in delivering data.

Our approach combines traffic shaping and power-aware scheduling to further reduce the energy consumption. Moreover, as the scheme is implemented at the local proxy, the delay in message exchange is low and traffic distortion is minimal.

6 Conclusion

In this work we aim to minimize the energy dissipation of mobile systems running online streaming media applications in a WLAN environment. We have analyzed the energy consumption of the wireless interface for receiving media streams over a wireless link. Note that a mobile client may be kept waiting while consuming energy for some time before it actually starts data reception. We show that the current power-saving mode (PSM) provided by the 802.11 standard is insufficient and point out traffic shaping alone does not give the best result for reducing communication energy.

We propose two power-aware scheduling schemes to enhance traffic shaping for additional energy savings. The algorithms build a transmission schedule at the local proxy server which informs the AP of the schedule that the streams should be delivered. The mobile clients simply follow the PSM specified by the 802.11 standards and do not need to be aware of the traffic scheduling scheme at the proxy. The first scheme we propose optimally minimizes the sum of finish times of all the clients running stream media applications in the WLAN. It does not require client side information and has been shown to be effective in energy minimization through simulation experiments. In the second scheme we take idle powers and residual battery capacities into consideration. We use the weighted sum of finish times as the performance metric. This has the effect of favoring the clients that are low in battery power and who idle power consumption rate is high. The optimal dynamic programming scheme we propose has pseudo-polynomial time complexity. We also present an efficient heuristic. Simulation results have shown that the heuristic achieves near-optimal performance and is able to attain significant improvement over using traffic shaping alone.

References

1. Stemm, M., Katz, R.H.: Measuring and reducing energy consumption of network interfaces in hand-held devices. IEICE Transactions on Communications, Special Issue on Mobile Computing **E80-B** (1997) 1125–1131
2. Shih, E., Bahl, P., Sinclair, M.J.: Wake on wireless: an event driven energy saving strategy for battery operated devices. (In: Proceedings of MOBICOM'02) 160–171
3. Gast, M.: 802.11 Wireless Networks: The Definitive Guide. O'Reilly (2002)
4. Bruno, J., E. G. Coffman, J., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Communications of ACM **17** (1974) 382–387
5. Brucker, P.: Scheduling Algorithms. 4th edn. Springer-Verlag (2003)
6. Garey, M.R., Johnson, D.S.: "Strong" NP-completeness results: motivation, examples, and implications. Journal of ACM **25** (1978) 499–508
7. Biyikoglu, E., Prabhakar, B., Gamal, A.E.: Energy-efficient packet transmission over a wireless link. IEEE/ACM Transactions on Networking **10** (2002) 487–499
8. Schurgers, C., Raghunathan, V., Srivastava, M.: Power management for energy-aware communication systems. ACM Transactions on Embedded Computing Systems **2** (2003)
9. Krashinsky, R., Balakrishnan, H.: Minimizing energy for wireless web access with bounded slowdown. (In: Proceedings of MOBICOM'02)
10. Zhang, F., Chanson, S.T.: Throughput and value maximization in wireless packet scheduling under energy and time constraints. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS). (2003) 324–334
11. Anastasi, G., Conti, M., Gregori, E., Passarella, A.: Performance comparison of power-saving strategies for mobile web access. Performance Evaluation **53** (2003) 273–294
12. Gundlach, M., Doster, S., Yan, H., Lowenthal, D.K., Watterson, S.A., Chandra, S.: Dynamic, power-aware scheduling for mobile clients using a transparent proxy. In: Proceedings of International Conference on Parallel Processing (ICPP). (2004)
13. Chandra, S.: Wireless network interface energy consumption: implications for popular streaming formats. Multimedia Systems **9** (2003) 185–201
14. Acquaviva, A., Simunic, T., Roy, S., Deolalikar, V.: Remote power control of wireless network interfaces. Journal of Embedded Computing (2004)

Encodings of Multicast Trees

Vijay Arya[1], Thierry Turletti[1], and Shivkumar Kalyanaraman[2]

[1] INRIA Sophia Antipolis, France
[2] Rensselaer Polytechnic Institute, NY USA
{vijay.arya, thierry.turletti}@sophia.inria.fr,
shivkuma@ecse.rpi.edu

Abstract. In this paper, we present efficient ways of encoding multicast trees. Multicast tree encodings provide a convenient way of performing stateless and explicit multicast routing in networks and overlays. We show the correspondence of multicast trees to theoretical tree data structures and give lower bounds on the number of bits needed to represent multicast trees. Our encodings can be used to represent multicast trees using both node identifiers and link indexes and are based on balanced parentheses representation of tree data structures. These encodings are almost space optimal and can be read and processed efficiently. We evaluate the length of these encodings on multicast trees in generated and real topologies.

1 Introduction

Current IP multicast routing protocols such as DVMRP [19] and PIM [8] install a per group state in the routers. To support these protocols, a router needs to maintain a multicast forwarding table entry corresponding to every multicast group whose distribution tree passes through the router. Since entries corresponding to different multicast groups cannot be aggregated, the forwarding table size grows with the number of active multicast groups, violating the stateless principle followed by routers. As the number of multicast groups increase, the cost and performance of routers are adversely affected. The state maintenance problem is further aggravated by the fact that even small multicast groups can introduce significant amount of state into the network. The amount of state introduced by a multicast group in the network depends on the size of its tree. The size of the multicast tree in turn is a function of both the relative locations of the source and various receivers, and the number of receivers itself. A small multicast group consisting of receivers which are spread uniformly in the Internet may introduce more state than a large multicast group concentrated in one region of the Internet. Thus, even a large number of small multicast groups may cause a state explosion. In order to reduce the state overhead at routers, one of the approaches followed is to move the state from the network to the packet, i.e., to use a representation of the multicast tree within every data packet. Each router can then function in a stateless manner and perform multicast forwarding

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 992–1004, 2005.

by reading and processing packet headers. To use this approach even in small to medium sized multicast groups, a compact encoding of multicast tree is required.

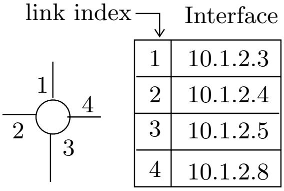
Due to deployment problems in Inter-domain multicast, several application level multicast protocols have been proposed [16, 21]. However, the problem of per group state maintenance remains even in application level multicast. Scribe [16] and Bayeux [21] are application level multicast protocols for Pastry and Tapestry overlays respectively. Pastry and Tapestry are large scale DHT overlays which allow interconnection and routing between millions of nodes in a scalable manner. To implement multicast in these overlays, both Scribe and Bayeux introduce a per group multicast state within overlay nodes which are part of the multicast distribution tree. Due to constraints of per group state maintenance, these overlays cannot support a large number of multicast groups (small or big). State maintenance is a serious issue in overlay nodes since they are constrained by memory and processing power (normally PCs/servers).

Reduction of per group state is just one of the several motivations for encoding Multicast trees. In Distributed Interactive Applications (DIAs) such as large scale virtual environments (e.g. multiplayer games) and distributed interactive simulations, participants act as senders and receivers in several multicast groups. Minimizing the control overhead due to group dynamics is a major design consideration here [10, 15]. In these applications, nodes are clustered according to communication needs and multicast distribution trees are constructed within the clusters. As nodes join and leave, they send control messages to each other to repair the multicast distribution trees. On average, the number of messages is linear to the number of nodes in the tree. Since there are a large number of multicast groups with frequent joins and leaves, significant control messages flow in the network. If the multicast trees are encoded within data packets, control overhead is reduced and on-the-fly tree repair is made possible at forwarding nodes [13].

Multicast tree encodings also provide a convenient way of choosing explicit multicast trees in networks and overlays. In overlay networks such as RON [1], OMNI [17], and Akamai, by embedding multicast trees within data packets, multicast traffic can be routed on specific links to perform load balancing or traffic engineering. Multicast tree encodings can also be used to communicate group keys needed for secure group communication [20] to specific receivers.

The above motivations notwithstanding, very little is known about optimal encodings of multicast trees. Indeed, there are two proposals which encode multicast trees non optimally [11, 13]. In this paper, we consider multicast trees which could occur at network or overlay level and show the correspondence of these trees to theoretical tree data structures. Using this correspondence, we give lower bounds on the number of bits needed to encode multicast trees and show efficient ways of encoding them.

Multicast tree encoding must satisfy two requirements. Firstly, the encoding must be of minimal length since the it is passed in *every* multicast data packet and processed by each forwarding node (router or overlay node). Secondly, the forwarding operation must consume minimal time. Each forwarding node pro-



link index	Interface
1	10.1.2.3
2	10.1.2.4
3	10.1.2.5
4	10.1.2.8

Fig. 1. Link Index concept: A router with its link indexes

cesses the encoding to determine the list of downstream nodes to whom the multicast packet must be forwarded. To minimize both the end-to-end packet delay and the per packet processing load on a forwarding node, the encoding must support the execution of this task efficiently.

The networks in which we consider multicast trees can be divided into two categories: (i) Networks which can *potentially* support forwarding based on *link Indexes.* The link index of a node with value i refers to its ith link. For example, the link index of a router refers to one of its interfaces. A router can potentially order its link interfaces and refer to each interface by its index in this ordering (Fig 1). The link index of an overlay node refers to one of its neighbors. Using a link index, a node can be instructed to forward a packet to one of its neighbors. This operation can potentially be supported in IP networks, static overlays such as OMNI, RON, Akamai, partially upgraded IP networks such as Bananas [9], etc where neighbors of a node do not change frequently. (In Bananas, nodes do forward unicast packets based on link indexes to support multipath-routing). (ii) In dynamic peer to peer networks (e.g. DHTs, unstructured p2ps) both the degree and the neighbors of nodes change frequently as nodes join and leave and hence forwarding based on link indexes cannot be supported. In these networks, node identifiers (e.g. IP addresses) of nodes need to be used to encode the multicast tree.

We show encoding of multicast trees using both link indexes and node identifiers. The tree encodings we propose are independent of underlying tree construction protocols and are based on *balanced parentheses representation of ordinal trees.* The rest of the paper is organized as follows. Section 2 places related work in context, Section 3 shows correspondence of multicast trees to tree data structures and Section 4 presents various tree encodings. Section 5 evaluates the length of these encodings on real and generated multicast trees and Section 6 concludes.

2 Related Work

The encoding of a multicast tree depends on whether it represents the *default* multicast tree supplied by unicast routing framework of the underlying network or whether it represents a *specific or explicit* tree. To represent a specific tree, the multicast tree must be encoded using either the node identifiers or link indexes of the entire tree. To choose the default tree, identities of receivers are sufficient. Default trees can be chosen in any network which supports unicast routing and their main application is multicast state reduction. On the other hand, explicitly representing trees provides the option of routing multicast traffic as desired which may be useful to create a multicast routing framework at application level, perform traffic engineering, or multicast state reduction.

In Xcast [7], a multicast tree is encoded by listing the IP addresses of receivers in the multicast tree. In IP networks, a packet can be routed to a destination node given its IP address. Due to this reason, explicitly listing the IP addresses of receivers of a multicast group suffices to represent the default multicast tree that exists between the source and the receivers. Forwarding routers perform a routing table lookup for each IP address in the packet and group them based on the common outgoing interface. The multicast packet is then forwarded on the interfaces, each packet containing the corresponding subset of IP addresses. Alternatively, if the last hop routers are allowed to store multicast state, the tree can be represented by listing the IP addresses of last hop routers instead of receivers (Xcast+). Xcast has two limitations. Since each IP address consumes 32 bits, Xcast encoding is suitable for multicast groups containing a small number of last hop routers (6-10 last hop routers need 24-40 bytes in IPv4). Secondly, each forwarding router needs to perform k IP lookups, where k is the number of last hop routers in its subtree. Large encodings require more IP lookups which are expensive.

The Xcast model is also suited to peer to peer DHTs such as Pastry and Tapestry, where packets are routed in the overlay using node identifiers. In Pastry, a node is represented using a 128 bit identifier. By representing multicast trees using receiver identifiers, the state overhead imposed by small multicast groups can be avoided in Scribe and Bayeux. The Xcast model is advantageous in DHTs since they guarantee routing in the presence of joins and leaves by intermediate peers.

In Linkcast [11], a multicast tree is encoded using the link indexes of links in the multicast tree. Each forwarding node reads *its* corresponding link indexes and forwards the multicast packet. The multicast tree which is encoded is a reverse path multicast tree constructed using join messages sent from last hop routers towards the source. Each router receiving the join message is expected to forward its identity along with the link index of the interface which received the join message. Thus the source can encode the multicast tree using link indexes. For forwarding, a pointer in the encoding points to the location of the current router in the encoding. Each router uses the pointer to read its outgoing link indexes and updates the pointer for the next hop routers. Although the number of links in the multicast tree is larger than the number of receivers, link indexes can be represented using fewer bits as compared to 32 bits for the IP address of a node. Also, expensive routing table lookups are avoided in Linkcast. We propose a new encoding for Linkcast which requires less space and two new encodings of multicast trees using link indexes.

In [13], authors propose two tree encodings to represent application level multicast trees occurring within clusters of DIAs. The multicast trees are encoded using IP addresses of participating nodes. Authors propose two encodings

- Per level header encoding and Preorder header encoding. In addition to IP addresses of nodes, these encodings spend significant number of additional bits. In a multicast tree of maximum degree d, the per-level header encoding spends $\log d$ bits per node and preorder header encoding spends approximately $\log n$ bits per node. We show an encoding using only 2 additional bits per node.

3 Multicast Trees and Tree Data Structures

Figure 2(i) shows a multicast tree occurring in an arbitrary network or overlay. The portion of the multicast tree which needs to be encoded is shown highlighted. The last hop routers, which serve one or more receivers are shown grayed. The link indexes corresponding to one of the nodes are shown. Figure 2(ii), (iii) and (iv) show the *Ordinal tree*, *Cardinal tree* and *Arbitrarily labeled tree* data structures corresponding to the multicast tree.

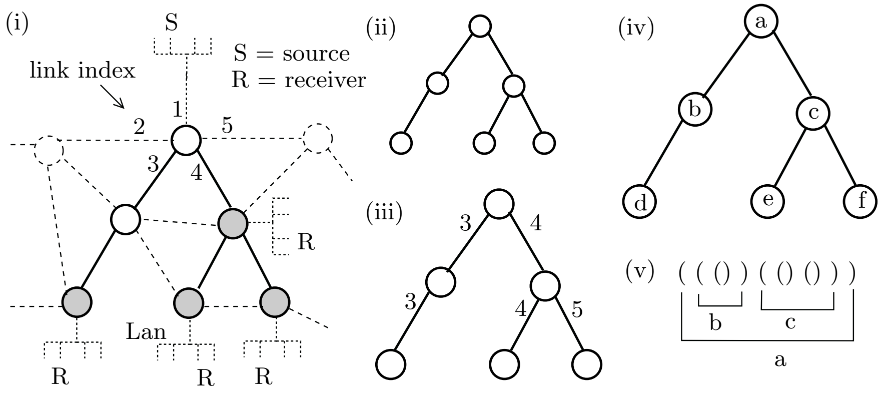


Fig. 2. (i) Multicast Tree in a Network (ii) Ordinal tree (iii) Cardinal tree (iv) Arbitrarily labeled tree (v) Balanced parentheses representation of ordinal tree

An ordinal tree is a rooted (a unique node is distinguished as the root), unlabeled tree in which each node has an arbitrary degree. A binary tree is an ordinal tree in which the maximum degree of a node is two. The number of ordinal trees on n nodes can be bounded by $O_n = \binom{2n+1}{n}/(2n+1)$. Thus, $\lg(O_n) \approx 2n$ is an information theoretic lower bound on the number of bits needed to represent ordinal trees (lg is $\log_2$). Every ordinal tree of n nodes can be represented using $2n$ balanced parentheses. Fig 2(v) shows the balanced parentheses representation of the ordinal tree in Fig 2(ii). This representation is obtained by a preorder (depth first) traversal of the tree, outputting a "(" while visiting a node for the first time and a matching ")" while visiting the node after visiting its subtree (The correspondence of some parentheses to nodes is shown). By representing a "(" by `1` and a ")" by `0`, balanced parentheses can be represented using $2n$ bits.

Figure 2(iii) shows the cardinal tree which results when a multicast tree is represented using link indexes. A cardinal tree of degree d is an unlabeled tree in which each node has d positions for an edge to a child (if a child is at the ith position, the corresponding link gets the label i). There are $C_n^d = \binom{dn+1}{n}/(dn+1)$ cardinal trees of degree d on n nodes. $\lg(C_n^d) \approx (\lg d + \lg e)n$ is an information theoretic lower bound on the number of bits needed to encode these trees. A multicast tree in which the maximum link index is d is a cardinal tree of degree d. Figure 2(iii) is a cardinal tree of degree 5. Thus, $(\lg d + \lg e)n$ is a lower bound on the number of bits needed to represent a multicast tree using link indexes.

Figure 2(iv) shows the arbitrarily labeled tree which results when a multicast tree is represented using node identifiers. Node Identifiers a,b, etc are essentially IP addresses. A labeled tree of n nodes is an ordinal tree in which each node is labeled from 1 to n. An arbitrarily labeled tree is an ordinal tree in which each node has a *unique* arbitrary label. If each node is represented using a k bit label, then the number of arbitrarily labeled trees on n nodes can be bounded by $A_n^k = \binom{2^k}{n}\ n!\ O_n$. Thus, $\lg(A_n^k) \approx kn + 2n$ is a lower bound on the number of bits needed to represent a multicast tree using node identifiers.

4 Multicast Tree Representations Using Link Indexes

For ease of explanation, we use the following notation. All nodes in the multicast tree have one incoming link (except the root) and zero or more outgoing links. A *terminal or leaf* node has zero outgoing links, a *relay* node has one outgoing link and a *branch* node has more than one outgoing links. In Fig 3(i), a, e, h and d are branch nodes, j, k, m, c, f and g are leaf nodes, and b, i and l are relay nodes. The links of a multicast tree can be divided into branch and relay links. All outgoing links of a branch node are branch links and all outgoing links of a relay node are relay links. For example, ac is a branch link and lm is a relay link. Let l denote the number of links and n denote the number of nodes, $n = l + 1$. Let $\bar{b}$ denote the number of branch links and $\bar{r}$ the number of relay links, $l = \bar{b} + \bar{r}$. Let b denote the number of branch nodes, r the number of relay nodes, and t the number of leaf nodes, $n = b + r + t$. For the tree in Fig 3(i), the preorder node traversal gives $a, b, e, h, j, k, i, l, m, c, d, f, g$ and the preorder link traversal gives $ab, be, eh, hj, hk, ei, il, lm, ac, ad, df, dg$.

4.1 Improvements to Linkcast Encoding

Linkcast consists of two encodings - SBM and DBM. These encodings consist of a series of elements. In SBM, each element is a link index or a pointer. A pointer is used to point to the representation of the child node in the encoding. In DBM, each element is either a link index, pointer or a node demarker. SBM requires $\bar{b}$ pointers. DBM reduces this to $\bar{b} - b$, but at the expense of introducing $l + 1$ node demarkers. Each node demarker requires at least two bits to be differentiated from both link indexes and pointers, consuming $2(l + 1)$ bits of space. Utilizing the concept of pointers, we now present a simplified encoding which is shorter than both SBM and DBM and refer to this encoding as Link+. Instead of using node demarkers, our approach is to distinguish between links that terminate at routers which perform forwarding and routers which do not, using one bit per link. One bit is used to distinguish between pointers and link indexes.

To construct Link+ encoding, nodes are visited in preorder. When a node is visited, a string of pointers and outgoing link indexes is outputted. If a node has d outgoing links, then $d-1$ pointers followed by d link indexes are outputted. The $d-1$ pointers point to the output strings of 2nd to dth child nodes. The output string of 1st child node occurs immediately after the output string of current node. Fig 3(ii) shows the Link+ encoding for the tree in Fig 3(i). For example, root node a has three outgoing links, thus 2 pointers followed by three link indexes are outputted. The output string of the first child node b occurs after the output string of a. The 1st pointer points to the output string of 2nd child node c (null). The 2nd pointer points to the output string of 3rd child node d. The first bit in each element is used to distinguish a pointer (`0`) from a link index (`1`). The second bit in each link index is used to distinguish links which terminate on leaf nodes (`0`) from those which do not (`1`).

Routing. In Link+, each node receives the pointer to its position in the encoding (as Linkcast). The root node receives a pointer to the first element in the encoding. Each forwarding node determines the (i) outgoing links to forward the packet (ii) positions of child nodes in the encoding in turn to be forwarded to the children. If the position of the current node starts with a pointer, then it is a branch node, else it is a relay node. If a branch node finds d pointers, it reads the subsequent $d+1$ elements to determine its outgoing links. A relay node receives the position of its outgoing link and its child is the next element. If the second bit of an outgoing link is 0, a null pointer is forwarded and thus a leaf node receives a null pointer. The forwarding operation at a node takes time proportional to its out degree. The detailed algorithm is given in [2].

In total, the encoding length of Link+ is $(S_l+2)l+(S_p+1)(\bar{b}-b)$ bits. S_l is the number of bits needed to represent the maximum link index in the tree. S_p is the number of bits needed to represent a pointer, $S_p = \lceil \lg(l+\bar{b}-b) \rceil$.

4.2 Encoding Based on Balanced Parentheses

We now present a link index encoding based on balanced parentheses representation. We refer to this encoding as Link*. Link* encoding consists of two parts (i) balanced parentheses representation of the tree (ii) preorder list of link indexes. The first encoding in Fig 3(iii) shows the encoding for the tree in Fig 3(i). The balanced parentheses representation is similar to the balanced parentheses representation of ordinal trees, except that each parenthesis now corresponds to the incoming link of a node, instead of the node itself. The encoding is constructed by a preorder traversal of the tree. When a link is visited for the first time, its link index and a ”(” are outputted. When a link is visited after visiting all links in its subtree, a ”)” is outputted. This results in $2l$ parentheses and l link indexes. In total, Link* requires $(S_l+2)l$ bits. S_l as before denotes the number of bits needed to represent the maximum link index.

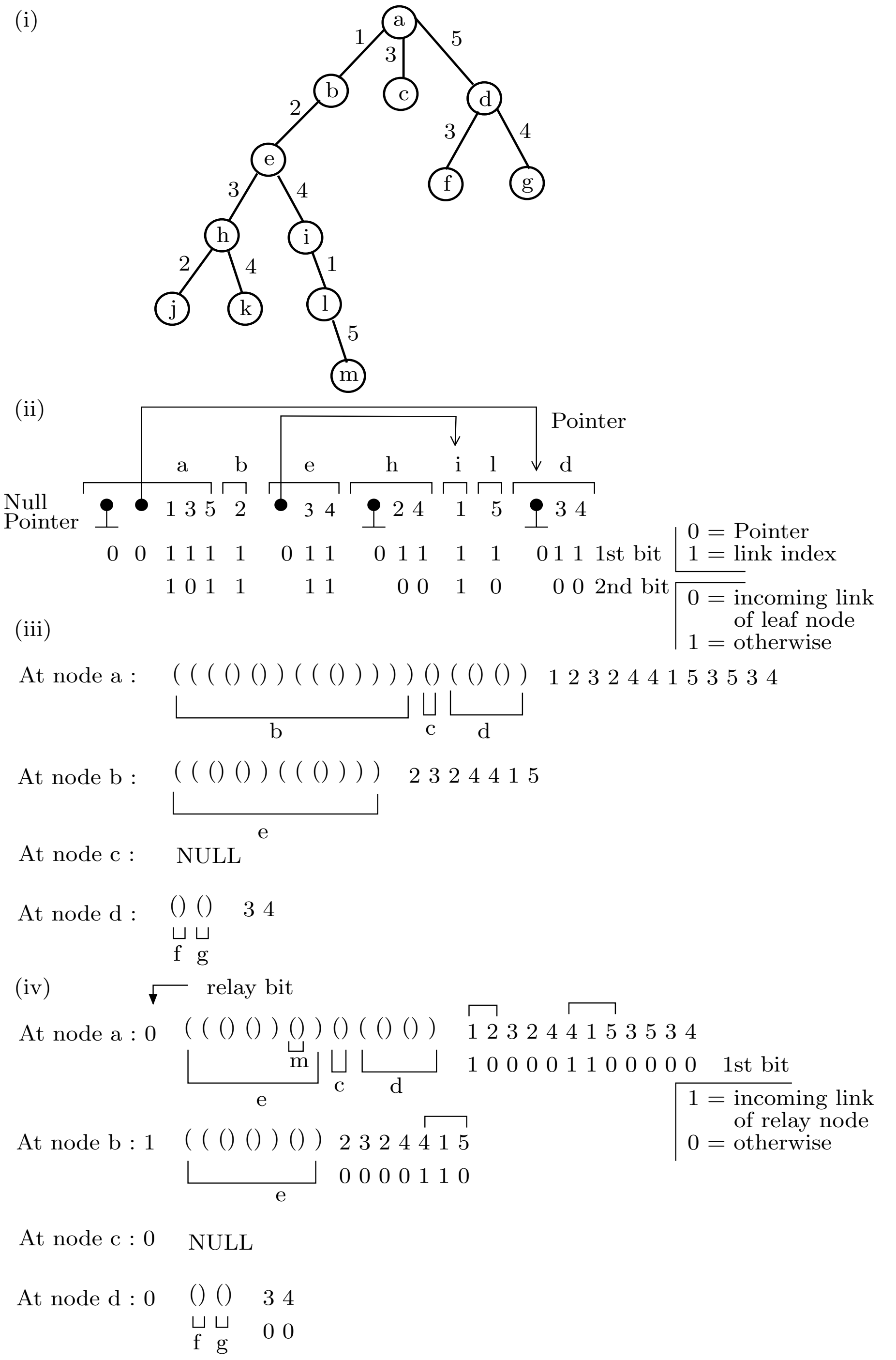

(i)
a
1
3
5
b
c
d
2
3
4
e
f
g
3
4
h
i
2
4
1
j
k
l
5
m
(ii)
Pointer
a b e h i l d
Null Pointer
1 3 5 2 3 4 2 4 1 5 3 4
0 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1st bit
1 0 1 1 1 1 0 0 1 0 0 0 2nd bit
0 = Pointer
1 = link index
0 = incoming link of leaf node
1 = otherwise
(iii)
At node a : (((() ()) ((())))) () (() ()) 1 2 3 2 4 4 1 5 3 5 3 4
b c d
At node b : ((() ()) ((()))) 2 3 2 4 4 1 5
e
At node c : NULL
At node d : () () 3 4
f g
(iv)
relay bit
At node a : 0 ((() ()) ()) () (() ()) 1 2 3 2 4 4 1 5 3 5 3 4
m
1 0 0 0 0 1 1 0 0 0 0 0 1st bit
e c d
1 = incoming link of relay node
0 = otherwise
At node b : 1 ((() ()) ()) 2 3 2 4 4 1 5
0 0 0 0 1 1 0
e
At node c : 0 NULL
At node d : 0 () () 3 4
f g 0 0

Fig. 3. (i) Multicast Tree (ii) Link+ encoding (iii) Link* encoding (iv) Link** encoding

Routing. In Link*, instead of passing a pointer to the position of a node in the encoding, the encoding itself is changed after each forwarding operation. Each node receives only the encoding of its subtree. Fig 3(iii) shows the encoding received by the root node a and the encodings it passes to child nodes b, c and d. A forwarding node reads the balanced parentheses once to determine the outgoing links and the encodings of subtrees to be passed to the child nodes. The ith open parenthesis "(" corresponds to the link index at the ith position in the list of link indexes within the encoding. For example, in Fig 3(iii), root node a reads its balanced parenthesis and determines that it has 3 children. The 1st "(" corresponds to child node b, the 9th "(" corresponds to c and the 10th "(" corresponds to d. Thus a's output links are at positions 1, 9 and 10 in the list of link indexes, i.e., link indexes 1,3 and 5 respectively. The detailed forwarding algorithm is given in [2]. The forwarding operation of a node takes time proportional to reading the balanced parenthesis representation of its subtree. (This encoding is similar to encoding of Munro, et al [3]. However, they consider efficient bit representations of balanced parentheses for very large trees, finding the parent of a node, size of subtrees, etc which are not needed for small and medium sized multicast trees. We use only the bare minimal representation of cardinal trees).

4.3 Improvements to Balanced Parentheses Representation

Several studies [5, 12, 6] have investigated the type of multicast trees which occur in the Internet. Trees occurring in the Internet are constrained by the underlying structure of the Internet. These trees have a large number of relay nodes compared to branch nodes. In Link*, other than the mandatory space of lS_l bits to represent the link indexes, 2 bits are spent for each link (for balanced parentheses). We now present a new encoding Link** which reduces the number of bits on those links which terminate on relay nodes and increases the number of bits on those links which terminate on branch nodes. In Link**, 3 bits are spent per incoming link of a branch node or leaf node, and only 1 bit is spent per incoming link of a relay node. In total, Link** requires $(S_l + 2)l + b + t - r$ bits. If $r > b + t$, the encoding spends less than 2 bits per link. In section 5, we shall see that for several trees in the Internet, the number of relay nodes is larger than the sum of branch and leaf nodes.

Link** encoding consists of three parts (i) relay bit (ii) balanced parentheses (iii) preorder list of link indexes. The encoding is similar to Link*. But it is obtained by encoding a virtual tree in which a path of consecutive relay nodes is treated as a single virtual link whose label is the concatenated list of its individual link indexes. Thus, the balanced parentheses represent only branch and leaf nodes. The first encoding in Fig 3(iv) shows the encoding for tree in Fig 3(i). For example, the path em is treated as a virtual link with label "415", and the path ae is treated as a virtual link with label "12". To determine the start and end of a virtual link, the first bit in each link index is used to distinguish links which terminate on intermediate relay nodes (`1`) from those which terminate on branch or leaf nodes (`0`). The relay bit is used for routing.

Routing. The forwarding operation is similar to Link*. Fig 3(iv) shows the encoding received by root node a and the encodings it passes to child nodes b, c and d. The ith open parenthesis "(" corresponds to the ith virtual link in the encoding. For example, in the encoding received by a, the 1st "(" corresponds to the 1st virtual link ae and the 5th "(" corresponds to the 5th virtual link em. During forwarding, a node reads the relay bit to check whether it is a relay node or not. The relay bit is `1` when the encoding is received by a relay node and `0` otherwise. The relay bit received by a node is simply the first bit taken from the link index of its incoming link. For example, when a performs forwarding, it sets the relay bit of child node b equal to the first bit of link index of link ab. In Link**, only branch nodes read the balanced parentheses and change it. The relay nodes read only the first link index to perform forwarding. The time complexity of forwarding is constant at relay nodes. The time complexity of forwarding at a branch node is proportional to reading the balanced parenthesis of its virtual subtree. The detailed forwarding algorithm is given in [2].

4.4 Representing Trees Using Node Identities

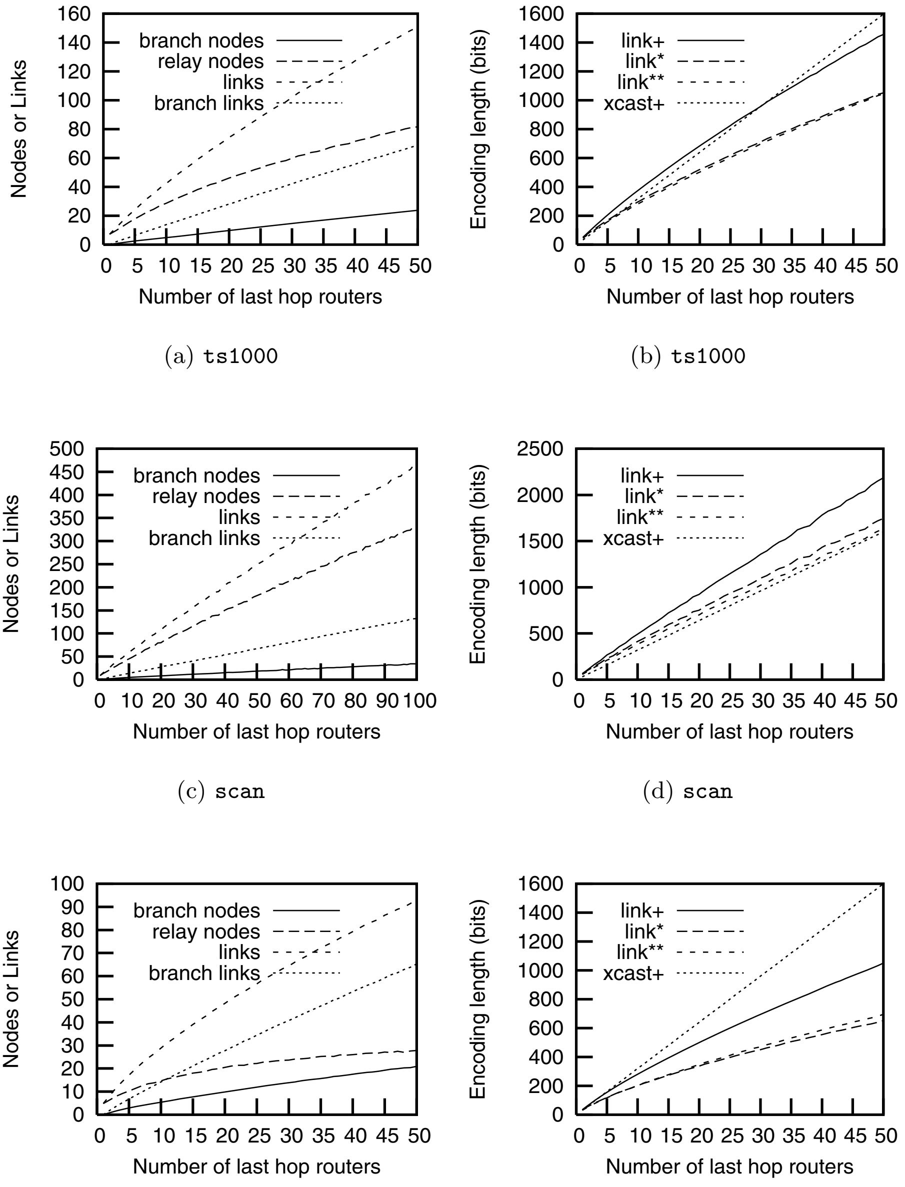
Both encodings Link* and Link** can be used *as is* by replacing link indexes of links by node identifiers of nodes which terminate on those links. Figure 4 shows the encoding Link* using node identifiers for tree Fig 3(i), as received by root node a. The forwarding algorithm remains the same, except that each forwarding node routes the packet to the corresponding next hop nodes instead of sending it on specific outgoing links [2].

$$\text{At node a :}\quad \underbrace{(\,(\,(\,()\,()\,)\,(\,(\,()\,)\,)\,)\,)}_{\text{b}}\;\underbrace{()}_{\text{c}}\;\underbrace{(\,()\,()\,)}_{\text{d}}\quad \text{b e h j k i l m c d f g}$$

Fig. 4. Link* encoding using nodes instead of links

5 Simulations

To evaluate these encodings, we conducted tests using various generated and real topologies of the Internet. Here, we present the results for three representative topologies - `ts1000`, `scan`, and `att`. `ts1000` is the transit-stub topology generated by GT-ITM [4] consisting of 1040 routers. `scan` is a partial map of the Internet collected by SCAN project [14] consisting of 284, 805 routers. `att` is the router level ISP topology of AT&T collected by Rocketfuel [18] consisting of 731 routers. For each of these topologies, we chose random routers as source and last hop routers and constructed shortest path multicast trees from source to last hop routers. To compute shortest paths, for `scan` and `att`, hop count metric was used and for `ts1000`, the generated weights were used. Figure 5 (a),(c) and (e) show the properties of multicast trees for the three topologies (each point is the average of 1000 simulations).



(a) ts1000

(b) ts1000

(c) scan

(d) scan

(e) att

(f) att

Fig. 5. Properties of multicast trees and their encoding lengths in various topologies

As observed, in all topologies the number of relay nodes is significantly larger than the number of branch nodes. However, the total number of branch links is significant and grows linearly with the number of last hop routers. Figure 5(b),(d) and (f) show the encoding lengths of Link+, Link*, Link**, and Xcast+. The Xcast+ encoding length is essentially 32 times the number of last hop routers. For link based encodings, the link index was represented using 5 bits (routers in the Internet rarely have degree larger than 32). The pointer in Link+ was represented using 8 bits. As seen, the Link+ encoding takes more space than both Link* and Link** to represent pointers corresponding to the branch links (Linkcast encodings take at least $l + 2$ more bits than Link+, for l links). For inter-domain topologies `ts1000` and `scan`, Link** encoding takes less space than Link* due to the presence of a large number of relay routers. For intra-domain topology `att`, as the number of last hop routers increase (> 25), Link* takes less space than Link** since the proportion of relay routers reduces compared to branch and last hop routers.

6 Conclusions

In this paper, we presented efficient ways of encoding specific or explicit multicast trees using link indexes and node identifiers. We presented an improved encoding Link+ and two new encodings Link* and Link** which consume space close to the minimum number of bits needed to represent multicast trees. These encodings can be used to represent multicast trees using either link indexes or node identifiers. We evaluated the space consumed by these encodings using shortest path multicast trees in real and generated topologies. These encodings can be used for various applications such as multicast state reduction, traffic engineering and application level multicast, and provide a feasible way of representing trees for small and medium sized multicast groups within data packets.

References

1. David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. The Case for Resilient Overlay Networks. In *8th Annual Workshop on Hot Topics in Operating Systems*, 2001.
2. Vijay Arya, Thierry Turletti, and Shivkumar Kalyanaraman. Encodings of Multicast Trees. Technical Report RR-5442, INRIA, Dec 2004.
3. David Benoit, Erik D. Demaine, J. Ian Munro, and Venkatesh Raman. Representing Trees of Higher Degree. In *International Workshop on Algorithms and Data Structures*, 1999.
4. Ken Calvert, Matt Doar, and Ellen W. Zegura. Modelling Internet Topology. *IEEE Communications Magazine*, 1997.
5. Robert C. Chalmers and Kevin C. Almeroth. On the topology of multicast trees. *IEEE/ACM Transactions on Networking*, 11(1):153–165, 2003.
6. Danny Dolev, Osnat Mokryn, and Yuval Shavitt. On Multicast Trees: Structure and Size Estimation. In *IEEE INFOCOM*, 2003.
7. Rick Boivie et al. Explicit Multicast (Xcast) Basic Specification, Internet Draft draft-ooms-xcast-basic-spec-06.txt, 2004.
8. S. Deering et al. The pim architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, pages 153–162, 1996.

9. Hema Tahilramani Kaur, Shivkumar Kalyanaraman, Andreas Weiss, Shifalika Kanwar, and Ayesha Gandhi. BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet. In *SIGCOMM FDNA Workshop*, 2003.
10. Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Paul T. Barham, and Steven Zeswitz. NPSNET: A Network Software Architecture For Large Scale Virtual Environments. In *Presence: Teleoperators and virtual Environments*, 1994.
11. Mozafar Bag Mohammadi, Siavash Samadian-Barzoki, and Nasser Yazdani. Linkcast: Fast and Scalable Multicast Routing Protocol. In *NETWORKING*, pages 1282–1287, 2004.
12. Graham Phillips, Scott Shenker, and Hongsuda Tangmunarunkit. Scaling of multicast trees: comments on the Chuang-Sirbu scaling law. In *Conference on Applications, technologies, architectures, and protocols for computer communication*, pages 41–51, 1999.
13. George Popescu and Zhen Liu. Stateless Application-Level Multicast for Dynamic Group Communication. In *IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, pages 20–28, 2004.
14. SCAN project. http://www.isi.edu/scan/mercator/maps.html.
15. J. Mark Pullen and David Wood. Network Technology for DIS. In *Proc. of IEEE*, pages 1156–1167, Aug 1995.
16. A. Rowstron, A-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Proc. of NGC*, 2001.
17. S.Y. Shi and J.S. Turner. Routing in overlay multicast networks. In *IEEE INFOCOM*, 2002.
18. Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP Topologies with Rocketfuel. In *SIGCOMM*, 2002.
19. D. Waitzman, C. Partridge, and S.E. Deering. Distance Vector Multicast Routing Protocol. RFC 1075, November 1988.
20. Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, 2000.
21. Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination. In *NOSSDAV*, 2001.

Efficient Bandwidth Guaranteed Restoration Algorithms for Multicast Connections

William Lau[1], Sanjay Jha[1], and Suman Banerjee[2]

[1] University of New South Wales, Sydney, NSW 2052, Australia
{wlau, sjha}@cse.unsw.edu.au
[2] University of Wisconsin-Madison, Madison, WI 53706, USA
suman@cs.wisc.edu

Abstract. This paper defines a new restoration strategy for provisioning bandwidth guaranteed recovery for multicast connections in presence of link failures. The new restoration strategy is formulated into a new Integer Linear Programming (ILP) algorithm and is compared with other existing restoration strategies. We also present a new heuristic algorithm based on this new restoration strategy. Results show that our heuristic algorithm performs competitively close to the ILP-based algorithm, and is more bandwidth efficient than other existing heuristic algorithms that are based on different restoration strategies.

1 Introduction

Networks are fast becoming the central medium for delivering rich multimedia contents to the clients that have the need and willingness to pay for the contents. With the trend of wide-spread broadband network access, content providers [1, 2] are already starting to deliver network-based multimedia service to the end users. There is also a emerging trend for content providers to use terrestrial networks as complement to the traditional satellite networks [3] for delivering contents to sites that their satellites cannot reach. There is a potential of using multicast solutions in terrestrial network in place of satellite network in the middle and long term due to the following factors: First, long disruptions to service due to bad weather are eliminated. Second, optical backbone bandwidth is growing at a much faster pace than satellite networks. Third, the accessibility and reachability of terrestrial networks are growing at a tremendous pace. We believe that these factors will be the main driving force for supporting multicast in backbone networks. However, existing work based on best effort routing [4, 5] and overlays [6, 7] will not be able to satisfy the stringent requirements of content service providers such as bandwidth guarantees, minimal disruption time, and ability to scale to large number of multicast connections. There is a need for traffic engineering capability that can satisfy these requirements.

The focus of our work is on the intra-domain multicast connections where the source node and the destination nodes reside in the same network domain. Each domain's source node can be considered as a Point-of-Presence (PoP) for

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1005–1017, 2005.

the content provider. Contents are delivered to the PoP for distribution within the domain. Inter-domain solutions for connecting the PoPs are outside the scope of this paper. We assume that the underlying network architecture is connection-oriented, such as MPLS [8] or GMPLS[9], which supports Point-to-Multi-Point (P2MP) [10] label switching delivering capability. This is required to allow the restoration strategy more control over the routing of the traffic in order to better utilize the links in the network. Another assumption is that only bandwidth recovery guarantees for single failures are considered because single failures are the most common type of network failures [11]. However, due to the space limitations, only single link failures are investigated in this paper. This paper proposes a new restoration strategy for provisioning bandwidth guaranteed recovery for multicast connections over mesh-based backbone networks. This restoration strategy is based on pre-computing the routes to take during failure and pre-allocate the resources required for the traffic during failure. This form of pre-planning allows bandwidth guaranteed recovery and also lowers the disruption time of the multicast connection.

We use the online pre-planning technique [12] where each connection request is pre-planned on-demand. That is, as the request is made to the network, it is pre-planned based on the current resource state of the network and does not need prior knowledge of other requests that have not been admitted into the network. This better suits on-demand multicast applications. We derived a new Integer Linear Programming (ILP) formulation based on this new restoration strategy and compare the results with ILP formulations for other existing restoration strategies. The results show that our restoration strategy requires significantly less bandwidth than the other strategies and has higher number of accepted connection requests. We also propose a new heuristic algorithm based on the new restoration strategy, and the results show that the heuristic algorithm performs competitively close to the ILP-based algorithm. Our heuristic algorithm is compared with other existing heuristic algorithms that use different restoration strategies, results show that our heuristic algorithm is more bandwidth efficient.

The rest of this paper is organized as follows. The related work is discussed in section 2 and then the new restoration strategy is presented in section 3. The experimental setup is described in section 4. The results for comparing between the different restoration strategies are presented and analyzed in section 5. In section 6, a new heuristic algorithm is introduced and the performance is evaluated in section 7. This paper concludes in section 8.

2 Related Work

Compared to unicast works, guaranteed bandwidth restoration strategies for multicast are limited. Most literatures focus on connectivity issues or probabilistic approaches. This section discusses the related algorithms.

Medard et al [13] propose an algorithm to compute service tree and a backup tree that is link (or node) redundant. Link redundant means that for a single link failure, all the nodes in the tree are still connected to at least one of the tree from the source. Thus this is a stronger requirement than required by a multicast tree which requires only the leaf nodes by protected.

Kodialam et al. [12] propose an approximation algorithm based on the line restoration strategy. Line restoration refers to the use of backup path segments that re-route the multicast traffic around the failed components and then back onto the multicast tree. For bandwidth efficiency, backup bandwidth can be shared between backup path segments. In this paper, we compare the performance differences between line restoration strategy and our tree restoration strategy using simulation experiments.

Singhal et al. [14] propose several new multicast restoration strategies using online planning. The best performing restoration strategy from their work is called Optimal Path-Pair (OPP). OPP uses disjoint path pairs between the source and each destination. Their results showed that OPP performs significantly better than link redundant trees [13]. However, their algorithm does not share backup bandwidth between backup trees that protect different service trees. This paper compares OPP to our proposed tree restoration strategy.

3 New Restoration Strategy

We propose a new restoration strategy based on the online pre-planning technique [12] where each connection request is pre-planned on-demand. Thus planning is made one request at a time and the objective of the restoration strategy is to minimize the bandwidth requirement for provisioning bandwidth guaranteed recovery for this requested multicast connection. The problem of finding the least cost multicast tree is a well-known NP-Hard problem (Steiner Tree), therefore, we separate the computation of the service multicast tree from the computation of the backup multicast tree. A service tree is used for delivering traffic during normal operation of the network while the backup tree is used to deliver traffic during network failures. For the service tree computation, we use an existing Integer Linear Programming (ILP) formulation [15] that calculates the optimal least cost multicast tree. Given a service tree, we propose a new restoration strategy that protects the service tree from single link failures.

Assuming single link failures, it is possible to define a set of failure scenarios with each scenario covering a different link failure. Each failure scenarios are considered to be independent of each other, as they are not expected to occur simultaneously. Given a service tree, we can identify the subset of failure scenarios that affects the service tree. For each of these failure scenarios, we can calculate a new multicast tree that is used to deliver traffic during this failure. Thus there are multiple backup trees protecting one service tree and the backup tree used during a failure depends on the state of the network (the failure). During a failure, the traffic is switched from the service tree to the appropriate backup tree. Thus the objective is to find the set of backup trees that protect the given service tree, and also to minimize the backup bandwidth required. To minimize the backup bandwidth requirement of the network, we use two techniques: Backup bandwidth sharing between backup trees belong in different failure scenarios, and Path Intermix [16] that allows the backup tree to reuse the bandwidth belonging to the service tree.

Suppose each link in the network reserves bandwidth used during normal operations (service bandwidth) and bandwidth used during failures (backup bandwidth). Establishing a service tree involves allocating more service bandwidth on the links used by the tree, and establishing a backup tree involves allocating more backup bandwidth. All service trees affected by a failure scenario are required to have a corresponding backup tree that protects the service tree from the failure. The total bandwidth required by the backup trees on link (i, j) (connecting from node i to node j) in failure scenario s is denoted by C^s_{ij}. Since failure scenarios are independent of each other thus the backup trees in different failure scenarios are not expected to be used at the same time. This allows the backup bandwidth between failure scenarios to overlap (share), hence we deduced the following relationship: $C_{ij} = \max C^s_{ij}, \forall s \in FS$

Where FS denotes the set of failure scenarios and C_{ij} is the total backup bandwidth allocated on link (i, j). Further, during a failure, the service bandwidth allocated to the service tree is idle since the traffic is switched to the backup tree. Thus there is an opportunity for the backup tree to reuse the service tree's bandwidth. The concept of using different type of bandwidth on different links in a backup tree is called Path Intermix [16]. It is likely that a link failure only affects only a small portion of the service tree, that is, only a subset of the destination nodes are affected. The sub-tree that connects the source to the unaffected destinations is used as the *skeleton* that forms the basis of the backup tree. The skeleton structure reuses the service bandwidth. To complete the backup tree, the skeleton tree is connected back to the affected destinations.

Table 1. Mathematical Notations

Symbol	**Definition**	**Symbol**	**Definition**
V	set of vertexes.	E	set of edges.
$G(V, E)$	The network graph	FS	set of failure scenarios.
s	A failure scenario in FS.	m	service tree.
FS_m	set of failures affecting m	h	source node.
d	A destination node in the request.	D	The set of destination nodes.
(i, j)	link connecting i to j.	b	requested bandwidth.
A_{ij}	bandwidth available in link.	C_{ij}	backup bandwidth in link.
C^s_{ij}	backup bandwidth required in (i, j) for s.	E^d_m	set of edges connecting source to d on m.
E_s	set of failed edges in s.	E_m	set of edges on m.
D_s	set of destination nodes affected by s.	τ^s_{ij}	cost for using (i, j) in s.

We call this the *Skeleton-Tree* restoration strategy. We formulate this restoration strategy into a new ILP problem that, given a service tree, finds the set of backup trees that protect the service tree with minimum additional backup bandwidth requirement to the network. The notations used is given in Table 1 The Skeleton-Tree Restoration ILP Formulation is described below. The objective is to find the minimum additional backup bandwidth required for the set of backup trees. The variable z_{ij} denotes the additional backup bandwidth required on link (i, j).

$$\min \left(\sum_{(i,j) \in E} z_{ij} \right) \tag{1}$$

$$\sum_j x_{ij}^{sd} - \sum_j x_{ji}^{sd} = 0, \text{ for } i \neq h, d \ , \forall d \in D_s \ , \forall s \in FS_m \tag{2}$$

$$\sum_j x_{hj}^{sd} - \sum_j x_{jh}^{sd} = 1, \forall d \in D_s \ , \forall s \in FS_m \tag{3}$$

$$\sum_j x_{dj}^{sd} - \sum_j x_{jd}^{sd} = -1, \forall d \in D_s \ , \forall s \in FS_m \tag{4}$$

$$\sum_j y_{ji}^{s} \leq 1, \forall i \in V - \{h\} \ , \forall s \in FS_m \tag{5}$$

$$\sum_j y_{jh}^{s} = 0, \forall s \in FS_m \tag{6}$$

$$x_{ij}^{sd} \leq y_{ij}^{s} \leq 1, \forall (i,j) \in E \ , \forall d \in D \ , \ \forall s \in FS_m \tag{7}$$

$$x_{ij}^{sd} = 1, \text{ If } (i,j) \in E_m^d \ , (i,j) \notin E_s \ , \forall d \in D - D_s \forall s \in FS_m \tag{8}$$

$$x_{ij}^{sd} = 0, \text{ If } (i,j) \notin E_m^d \ , (i,j) \notin E_s \ , \forall d \in D - D_s \forall s \in FS_m \tag{9}$$

$$z_{ij} \geq \tau_{ij}^{s}(y_{ij}^{s}), \forall (i,j) \in E \ , \ \forall s \in FS_m \tag{10}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \ . \ x_{ij}^{sd}, y_{ij}^{s} \in \{0, 1\} \ . \ z_{ij} \geq 0 \tag{11}$$

The link capacity constraints and the non-zero additional backup bandwidth constraints are defined in equation 11.For each failure, we find a multicast backup tree that connects to all the destinations. To create a multicast tree, we first establish a path between the source and each of the destinations affected in the failure scenario. Equations 2, 3, and 4 are for flow conservation to establish the paths. The variable x_{ij}^{sd} determines whether link (i, j) is used on the path to destination d for backup tree in failure scenario s. Variable y_{ij}^{s} determines whether link (i, j) is used on the backup tree in failure scenario s. Equation 7 forces the backup tree to include the links used in the individual paths. Equations 5 and 6 enforce the tree constraint where there can only be one incoming links used for each node while for the source node, no incoming links are used. These equations are necessary to prevent loops cause by links that can be used with zero cost. Skeleton-Tree Restoration forces the unaffected part of the service tree to be reused. Equations 8 and 9 enforces this requirement. The cost for using link (i, j) in failure scenario s is defined by τ_{ij}^{s}.

$$\tau_{ij}^{s} = \begin{cases} 0, & \text{if } (i,j) \notin E_s \text{ and } ((i,j) \in E_m \text{ or } C_{ij}^{s} + b \leq C_{ij}) \\ b - C_{ij} + C_{ij}^{s}, & \text{if } (i,j) \notin E_s \text{ and } C_{ij}^{s} + b > C_{ij} \\ & \text{ and } A_{ij} \geq b - C_{ij} + C_{ij}^{s} \\ \infty, & \text{otherwise.} \end{cases}$$

The first case is when the link can be used with zero cost because of path-intermixed or there is enough backup bandwidth for sharing. The second case occurs when there is not enough backup bandwidth on cover the all of the requested bandwidth. Thus the link cost is the additional bandwidth that needs to be allocated to cover the whole request. The final case occurs if the link being considered is down or there is not enough available bandwidth on this link to satisfy the request.

Backup trees for different failure scenarios can share backup bandwidth with each other thus equation 10 allows the backup bandwidth to overlap. The additional backup bandwidth required on each link is thus the maximum of the additional backup bandwidth required by the backup trees being calculated.

4 Experimental Setup

For the simulations, two network topologies are used and are shown in Figures 1 and 2. Network1 and Network2 [17] are networks taken from practical topologies. Without loss of generality in all networks, all links are bi-directional and the link weights are set to 1. For all experiments, the results are collected from the average of results from 10 different request sets. Each request set has a 1000 multicast connection setup requests. Requests are randomly generated and have uniform bandwidth demand of one unit each. The number of destinations per request is uniform within a request set. The number of destinations is varied across simulation runs. The number of destinations will vary from 2 to 8 in increment steps of 2. When the number of destinations reaches 10 or more, the simulation run-time for each request set goes for over 48 hours for the Skeleton-Tree ILP algorithm. Thus we limit the number of destinations to 8. The ILP formulations in the experiments are solved using CPLEX 8.1 and the heuristics algorithms are implemented using C++. The experiments are executed on a PIII 1Ghz Linux Server with 1GB of RAM.

There are two types of experiments, unlimited link capacity scenario, and limited link capacity scenario. Unlimited capacity experiments are used to show the bandwidth efficiency of the algorithms. This is important when the multicast connections share the network with other traffic. Limited capacity experiments show how much requests can be admitted in the network given limited resources allocated for provisioning multicast connections. For the limited link capacity

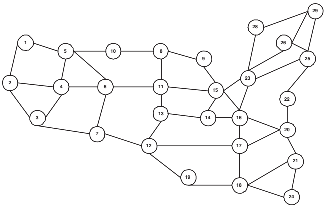


Fig. 1. Network1: US Long-Distance

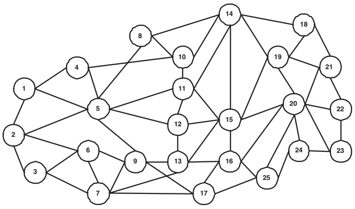


Fig. 2. Network2: Toronto Metropolitan

scenario all the links in the network will have the same capacity for simplicity. The number of destinations is fixed to 6 in the limited capacity experiments. The link capacity for each network topology is different due to the size and density of the network. Network1's link capacity varies from 0 to 450 units with incremental steps of 50 units. Network2's The link capacity varies from 0 to 250 units with incremental steps of 50 units.

Two metrics are used for analysis of results: Total bandwidth, and Number of Requests Accepted. Total bandwidth refers to the sum of the bandwidth required by all the service tree and all the backup paths/trees. Number of Requests Accepted is the number of requests that can be admitted into the network within the resource constraints. A request is blocked if the algorithm cannot find a feasible service tree or the set of feasible backup paths/trees required.

5 Simulation Results Part I

The first set of simulation results compare between three solutions: Optimal Path-Pair [14], Line Restoration [15], and Skeleton-Tree Restoration. Figures 3-4 show the total bandwidth requirements from the unlimited capacity experiments. Simulations for Skeleton-Tree Restoration only go up to multicast group size of 8 for Network1 and Network2. Going beyond group size of 8 increases the computation time exponentially. Results show that Skeleton-Tree Restoration has the lowest total bandwidth requirement, and is up to 10% less than Line Restoration in Network1 and 6% less in Network2. Note that Skeleton-Tree Restoration uses the same algorithm to compute the service tree as Line Restoration, thus the difference between the two solutions is the restoration strategy used. Compared with Skeleton-Tree Restoration, Line Restoration uses up to 26% more backup bandwidth in Network1 and 21% more in Network2. Thus Skeleton-Tree Restoration is a more bandwidth efficient strategy as compared with Line Restoration.

Line Restoration retains multicast efficiency by restoring the service tree and maximize reuses of the service bandwidth, however, it does not give the restoration algorithm enough freedom to spread out the backup bandwidth. The line segment will always be restricted to be near the link failure. Skeleton-Tree Restoration gives the algorithm complete freedom to choose the way the affected destinations join back the skeleton tree. Comparing Skeleton-Tree Restoration with Optimal Path-Pair, Skeleton Restoration uses up to 31% less total bandwidth in Network1 and 40% less in Network2. The difference is from the more efficient service tree algorithm used in Skeleton-Tree Restoration and also the efficiency gained from backup bandwidth sharing between backup trees.

Figure 5-6 show the number of requests accepted from the limited capacity experiments. Results show that Skeleton-Tree Restoration and Line Restoration perform within 1% of each other in Network1. In Network2, Skeleton-Tree Restoration accepts up to 4% more requests than Line Restoration. The reason why these two algorithms perform so close to each other is because the main factor contributing to the blockings is the service tree algorithm. That is, nearly all the blockings is from failing to find a feasible service tree. Although the backup

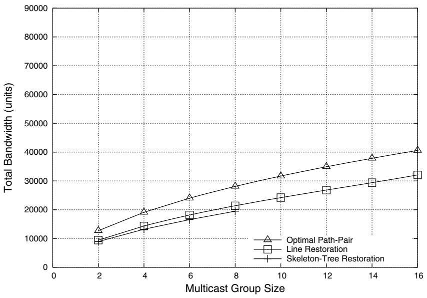


Fig. 3. Total Bandwidth Requirement for Network1

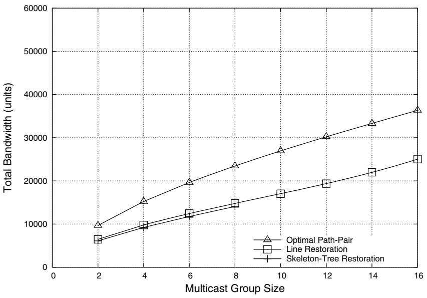


Fig. 4. Total Bandwidth Requirement for Network2

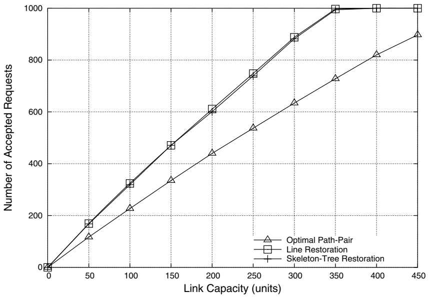


Fig. 5. Number of Request Accepted for Network1

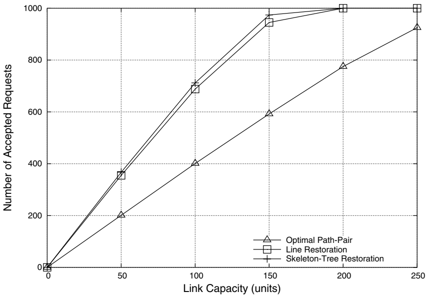


Fig. 6. Number of Request Accepted for Network2

bandwidth is minimized by the strategies, the service tree algorithm will eventually utilize the resources of the links in the congested areas (hot-spots). Thus the links on the hot-spots will always be fully utilized under all the algorithms. Note that finding a feasible service multicast tree is much harder than a unicast service path as the number of links required to set up a multicast tree is significantly higher (need to connect to 6 destinations in the simulations). Thus multicast service tree algorithm is observed to impose a higher impact on blockings than the backup restoration strategies because the service bandwidth contributes to most of the total bandwidth requirement. Comparing Skeleton-Tree Restoration with Optimal Path-Pair, Skeleton-Tree Restoration has up to 45% more accepted requests in Network1 and 103% more in Network2. The bandwidth efficiency of Skeleton-Tree Restoration allows substantially more requests to be accepted.

6 Heuristic Restoration Algorithm

Integer Linear Programming based algorithms are computationally expensive and are known to have exponential worst case times. The Skeleton-Tree ILP algorithm can take hours to compute for each set of 1000 requests with multicast group size of 8, and can go beyond 48 hours for group sizes of 10 and over. This is definitely not desirable for online-based solutions that require on-demand response time. Therefore, we propose an approximation algorithm for the Skeleton-Tree ILP algorithm. This algorithm is called *Approximate Multicast Restoration Algorithm* (AMRA) described in Algorithm 1. The same notation is used as shown in Table 1, however, we use l to represent a link rather than (i, j).

```
Data    : Network Graph: $G(V, E)$; Failure scenario set: $FS$; Service tree: $t$;
Result  : New Network Graph: $G(V, E)$; Backup tree set: $BT$; Failure scenario set: $FS$;
Initialize $BP = \emptyset$;
Find the set of failures $FS_t$ affecting the Service tree;
for each $s \in FS_t$ do
    Find the set of failed links $E_s$ in the current failure scenario;
    Remove all failed links $l \in E_s$ from $G(V, E)$;
    Find the skeleton of the service tree $tree_{skeleton}$;
    Recompute the network links cost $\tau_l^s$;
    Find the list of affected destinations $D_s$;
    for each $d \in D_s$ do
        If $d$ is already traversed in $tree_{skeleton}$ then skip to next destination;
        Find the shortest path $p_d$ from any node on $tree_{skeleton}$ to $d$;
        $tree_{skeleton} = p_d \cup tree_{skeleton}$;
        For all links used in $p_d$, change the link cost to $\infty$;
    end
    Add $tree_{skeleton}$ to $BT$;
    Update $G(V, E)$ with the new $tree_{skeleton}$;
    Insert failed links back into $G(V, E)$;
end
```

Algorithm 1: Approximate Multicast Restoration Algorithm

The input to the algorithm is the network resource state $G(V, E)$, the list of failure scenarios FS, and the service tree t. The expected output of the algorithm is the set of backup trees BT, and the updated network resource state. First, we find the list of failure scenarios that affect the service tree FS_t. For each of the failure scenarios, we remove the failed links (E_s) and then we calculate a backup tree that restores traffic to all destinations. The logic is to first form the skeleton of the backup tree from the unaffected portion of the service tree (the links that are used to connect the source to the unaffected destinations). We then find the list of affected destinations D_s. The special case of the skeleton tree is when all the destinations are affected thus the skeleton tree only includes the source node. We then go through each affected destination sequentially and find the least cost path that connects the skeleton tree back to the destination node. This path is then added to the skeleton tree and the links used in the path will now have infinite cost (prevent loops). This process continues until all the destinations are connected to the skeleton tree. This skeleton tree then becomes the new backup tree. The algorithm ends when we find the set of backup trees that protect the service tree from all failure scenarios.

```
Data     : Network Graph: $G(V, E)$; Skeleton tree: $t$; destination: $d$;
Result   : Least cost path: $p$;
$path_{min} = 0$;
for each $v \in V_t$ do
    Find the shortest path $p_{new}$ from $v$ to $d$;
    if cost of $p_{new} < path_{min}$ then
        $p = p_{new}$;
    end
end
```

Algorithm 2: Shortest Path to Skeleton Tree

To find the shortest path from the skeleton-tree to a destination, we use the algorithm described in Algorithm 2. This is essentially just a loop that goes through all the nodes in the skeleton tree (V_t) and finds the shortest path from the nodes to the destination. The shortest path with the least cost is returned. The cost for using link l during failure s is given by τ_l^s:

$$\tau_l^s = \begin{cases} \infty, & \text{if } l \in E_{skeleton} \\ 0, & \text{if } l \notin E_s \text{ and } l \in E_m \text{ and } l \notin E_{skeleton} \\ 0, & \text{if } l \notin E_s \text{ and } C_l^s + b \leq C_l \\ b - C_l + C_l^s, & \text{if } l \notin E_s \text{ and } C_l^s + b > C_l \\ & \text{ and } A_l \geq b - C_l + C_l^s \text{ and } l \notin E_{skeleton} \\ \infty, & \text{otherwise.} \end{cases}$$

The first case is for preventing the links ($l \in E_{skeleton}$) already on the skeleton tree from being re-used. This stops loops formation. The second case is for links that belong to the service tree that can be used for path intermix. These links must not be part of the current skeleton tree and is not one of the failed links. Path intermix allows these links to be used without additional cost. The third case is when there is enough backup bandwidth to cover the request in addition to what has already been used by this failure scenario. The fourth case is where the backup bandwidth can only cover part of the requested bandwidth. Thus the cost for using this link will be the additional backup bandwidth needed to cover the rest of the requested bandwidth. The final case is when sufficient bandwidth is not available on the link to cover the additional bandwidth requirement.

The time-complexity of AMRA is $O(|FS_t|(|D||V|^2 \log |V| + |E|))$ (shortest path $O(|V| \log |V|)$).

7 Simulation Results Part II

ARMA approximates the Skeleton-Tree ILP Restoration algorithm but does not compute the service tree. We use an existing approximation algorithm (Nearest Neighbor First) [12] for calculating the service tree. Both algorithms are polynomial-time thus the combination is a polynomial-time solution.

Simulations in this section compare our proposed heuristic algorithm with the ILP based algorithm and another existing heuristic algorithm: Skeleton-Tree

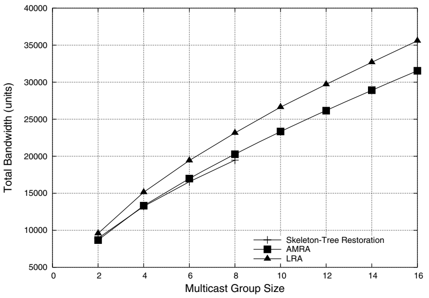


Fig. 7. Total Bandwidth Requirement for Network1

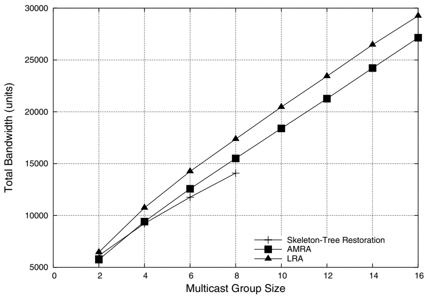


Fig. 8. Total Bandwidth Requirement for Network2

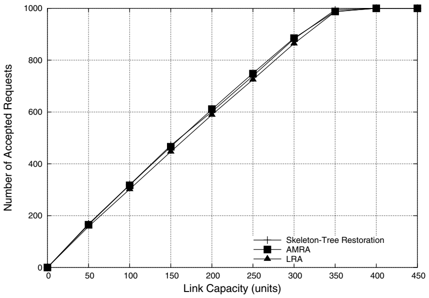


Fig. 9. Number of Request Accepted for Network1

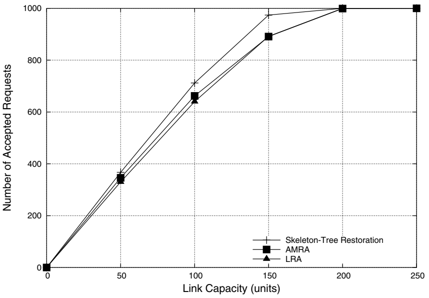


Fig. 10. Number of Request Accepted for Network2

Restoration ILP, AMRA, and Line Restoration Approximation (LRA) [12]. The first set of simulations is for unlimited link capacity networks with varying multicast group sizes. The total bandwidth requirement results are presented in Figures 7-8. Compared with Skeleton-Tree Restoration, AMRA requires up to 4% more bandwidth in Network1 and 10% more in Network2. The difference between the two solutions is mainly from the higher service bandwidth requirement. This is because Skeleton-Tree Restoration uses an ILP algorithm that finds the optimal least cost service tree whereas AMRA uses an approximation algorithm only. Comparing with the other heuristic solution, LRA requires up to 15% more bandwidth than AMRA in Network1 and 14% more in Network2. Since AMRA and LRA use the same service tree algorithm, the results show the backup bandwidth efficiency of the skeleton-tree restoration strategy as compared to line restoration strategy.

The second set of simulations is for limited capacity experiments where the multicast group size is fixed to 6 destinations while the link capacity varies. The results are shown in Figures 9-10. The results show that the three solutions perform very similarly in terms of the acceptance rate. AMRA performs about the same as Skeleton-Tree Restoration for Network1 (within 1%). For Network2, the acceptance rate for AMRA reaches within 9% of Skeleton-Tree Restoration. Compared with LRA, AMRA has up to 4% higher acceptance rate in Network1 and up to 5% higher in Network2. The main cause (nearly all) of request blocks for the algorithms is from failing to find a feasible service tree. Skeleton-Tree Restoration uses an ILP algorithm that will always find a feasible service tree if one exists. Thus they have slightly more acceptance rate than the heuristic algorithms, but the hot-spots cause all the solutions to block most requests at network saturation point.

8 Conclusion

In this paper, we proposed a full ILP-based solutions (Skeleton-Tree Restoration) and a complete heuristics solution (AMRA) based on a new restoration strategy. Simulation results show that using any of these solutions will improve bandwidth efficiency substantially over other existing solutions (Optimal Path-Pair and LRA). Bandwidth efficiency for AMRA is within 10% of the full ILP-based solution and the request acceptance rate is within 12%. The heuristics solution has polynomial-time complexity and is more scalable to larger networks and multicast group sizes than the ILP-based solution.

References

1. FastWeb: TV Over FastWeb. http://www.fastweb.it/ (2004)
2. now.com.hk: Pay TV Over Broadband. http://www.now.com.hk/ (2004)
3. TV, S.P.: Terrestial Broadcast. http://www.skyperfectv.co.jp/skycom/e/ (2004)
4. Moy, J.: Multicast Extensions to OSPF. IETF RFC 1584 (1994)
5. Ballardie, A.: Core Based Trees (CBT). IETF RFC 2201 (1997)
6. Chu, Y., Rao, S., Seshan, S., Zhang, H.: Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In: Proceedings of ACM SIGCOMM, San Diego, USA (2001)
7. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S.: Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications. In: Proceedings of IEEE INFOCOM, San Francisco, USA (2003)
8. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. IETF RFC 3031 (2001)
9. Berger, L.: Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description. IETF RFC 3471 (2003)
10. Yasukawa, S.: Requirements for Point to Multipoint extension to RSVP-TE. IETF Internet Draft draft-ietf-mpls-p2mp-requirement-01.txt (2004)
11. Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C., Diot, C.: Characterization of Failures in an IP Backbone. In: Proceedings of IEEE INFOCOM, Hong Kong, China (2004)
12. Kodialam, M., Lakshman, T.: Dynamic Routing of Bandwidth Guaranteed Multicasts with Failure Backup. In: Proceedings of IEEE ICNP, Paris, France (2002)
13. Medard, M., Finn, S., Barry, R., Gallenger, R.: Redundant Trees for Preplanned Recovery in Arbitary Vertex-Redundant or Edge-Redundant Graphs. IEEE/ACM Transactions on Networking **7(5)** (1999)

14. Singhal, N., Sahasrabuddhe, L., Mukherjee, B.: Provisioning of Survivable Multicast Sessions Against Single Link Failures in Optical WDM Mesh Networks. IEEE Journal of Lightwave Technology **21(11)** (2003)
15. Lau, W., Jha, S.: Multicast Resilient Connections with Quality of Service Guarantees. UNSW Technical Report 0408 (2004)
16. Lau, W., Jha, S.: Failure-Oriented Path Restoration Algorithm for Survivable Networks. eTransactions on Network Service and Management (IEEE Communications Society) **1(1)** (2004)
17. Xiong, Y., Mason, L.: Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks. IEEE/ACM Transactions on Networking **7(1)** (1999)

Scheduling Uplink Bandwidth in Application-Layer Multicast Trees

Sridhar Srinivasan and Ellen Zegura

Networking and Telecommunications Group, College of Computing,
Georgia Institute of Technology, Atlanta, GA 30332, USA
{sridhar, ewz}@cc.gatech.edu

Abstract. Many applications can benefit from the use of multicast to distribute content efficiently. Due to the limited deployment of network-layer multicast, several application-layer multicast schemes have been proposed. In these schemes, the nodes in the multicast tree are end systems which are typically connected to the network by a single access link. Transmissions to the children of a node in the multicast tree have to share this single uplink, a factor largely ignored by previous work.In this work, we examine the effect of access link scheduling on the latency of content delivery in a multicast tree. Specifically, we examine the general case where multiple packets (comprising a *block* of data) are sent to each child in turn. We provide an analytical relation to compute the latency at a node in the multicast tree and show the relationship to the packet size and block size used to transfer data. We propose heuristics for tree construction which take link serialization into account. We evaluate this effect using simulations and show that using larger block sizes to transfer data can reduce the average finish time of the nodes in the multicast tree at the expense of slightly increased variance.

Keywords: Peer-to-peer networks, Multicast.

1 Introduction

Application-layer multicast [6, 10, 1, 9, 5] has been proposed as a viable method for deploying large-scale multicast on the Internet. Unlike network-layer multicast, in most proposals for application-layer multicast, the nodes that form the multicast tree are end hosts. These end hosts are responsible for creating and maintaining the multicast tree and also forwarding the data to their children in the tree. Applications that benefit from the use of application-layer multicast include media streaming, multi-player games, conferencing and file or content distribution.

Important metrics for these applications are the delay and jitter experienced during data transfer. In the case of file distribution applications, the average time to obtain the file is also an important requirement. For this reason most application-layer multicast schemes concentrate on creating multicast trees with low latency paths.

The end hosts that form the multicast tree are often connected to the rest of the Internet using a single access link such as a DSL or cable modem line [14]. The single access link is a shared resource that must be scheduled among the children of the node. This scheduling can affect the time taken to transfer data to the child nodes. As a simple

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1018–1030, 2005.

example, consider a case in which a source node r with a single access link to the rest of the network, transfers a block of data to its k children. Let us further assume that this block is composed of several packets. Consider two different simple means of scheduling access to the link: in the first scheme, the block is sent a packet-at-a-time to each child in turn and in the second, the entire block is sent to a single child at a time. The methods of delivery are illustrated in Figures 1 and 2 respectively. The figures show the delivery of a block of data composed of four packets from a source node to its k child nodes. In the packet-at-a-time case, the finish times of all the children are nearly equal, while in the block-at-a-time method, the 1st child has a finish time of T, the second a finish time of $2 * T$ and so on. From the figures, we see that in the packet-at-a-time case, all children get the block at almost the same time while in the block-at-a-time case, some of the children get the block much earlier. It can be shown that the average finish time in the latter case is lower.

To the best of our knowledge, there has been no work examining the effect of this link sharing on the data delivery of application-layer multicast trees.[1] In this work, we analyze the effect of this link sharing and we demonstrate a simple technique, called Time Division Streaming, to exploit this sharing to reduce the average time to transfer data. We provide analysis of TDS using a simple network model. We then show how the construction of multicast trees can take advantage of TDS and propose heuristics for tree construction.Our results show that sending larger blocks of data in multicast trees constructed using our heuristic can provide a substantial improvement in the average finish time of the nodes at the expense of some increase in the maximum finish times of some nodes. We examine the tradeoff in our evaluation of TDS.

The rest of the paper is organized as follows: In Section 2, we describe the link sharing in detail. In Section 3, we develop the relation to compute the latency to any host in an end-system multicast tree. We then present algorithms and heuristics to create TDS trees in Section 4. We evaluate these TDS trees in Section 5 and discuss related work in Section 6. We conclude with some discussion of the results in Section 7.

2 Time Division Streaming

We present our work in the context of a data delivery application such as an audio or video stream that requires a minimum data rate or bandwidth of b bits per second (bps). We note that this data rate can be achieved either by small packets delivered at regular intervals or by sending a larger block of data, such as a block containing a few seconds of content, at the beginning of a time period such that the effective data rate over the time period is greater than or equal to b. Thus, in our discussion, we use the term “block” of data to imply the transfer of one or more back-to-back packets of data between nodes in the multicast tree.

[1] The authors in [3] point out that the single access link imposes the constraint that the data intended for children of this node must be serialized at the link though they do not consider the problem of scheduling the access to the link.

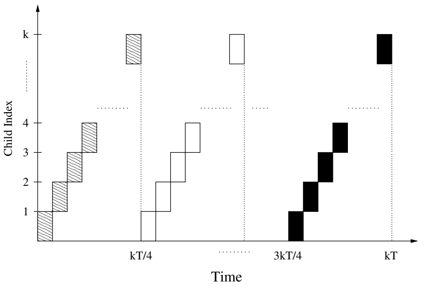


Fig. 1. A block comprising of four packets being sent to k children a packet at a time

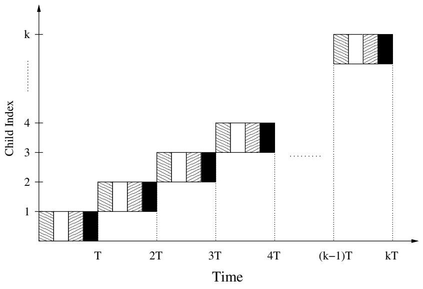


Fig. 2. A block comprising of four packets being sent to k children a block at a time

We now present a scheme to exploit the serialization of packets at the access link. We call this mode of transfer of data Time Division Streaming or TDS. The idea behind this mode of data transfer is essentially to use the available upload bandwidth of a node in a manner similar to Time Division Multiplexing (TDM) of a communication channel, hence the name. In this mode, a node will send a block of data, composed of several packets, to only one of its children, utilizing all of its upload bandwidth for that transfer[2]. Once the transfer is complete, the node sends the block to the next child in order and so on.

Effect of TDS on a Single Node. To illustrate the effect of link scheduling, we revisit our initial example shown in Figures 1 and 2. The figures show a scenario in which we do not consider the propagation delay to the nodes. Figure 1 illustrates the case where the block of four packets is sent as a packet to each child in turn. The average time to finish receiving the entire block for the children is $3kT/4 + (T/4)(k+1)/2$. Figure 2 shows the case in which the entire block is sent to each child. The average finish time in this case is $T(k+1)/2$. The gain in the average finish time for this works out to $(k-1)3T/8$ which is greater than zero for more than one child, i.e., $k > 1$. More generally, if we let n be the number of packets in a block, s be the size of a packet in bits and B be the uplink bandwidth at the node r, the average finish time, while sending a packet to each child in turn, is $(n-1)ks/B + (s/B)(k+1)/2$ whereas it is $(ns/B)(k+1)/2$ when sending a block at a time. The gain in average finish time over all the nodes is $(s/B)(n-1)(k-1)/2$. It is simple to conclude from this analysis that larger blocks are better at reducing the average finish time.

If we incorporate the propagation delay of the links connecting the nodes, the analysis remains unaffected as the propagation delay remains unchanged in both cases with only the transmission delay changing due to the use of larger blocks.

Effect of TDS in a Multicast Tree. To evaluate the effect of using TDS in a multicast tree, consider an interior node r in the tree. If the node has k children, each the root of a subtree, and the children are numbered from 1 to k, consider the finish time of

[2] For this work, we assume that the transport protocol used by the application allows for blocks of data to be sent in packets that are back-to-back on the connection.

the first child. If this child is the first to receive a block from r, its finish time is ns/B when n packets are sent as a block as opposed to $(n-1)ks/B + s/B$ when a single packet is sent to each child in turn. Irrespective of the scheduling in any other node in this subtree, the finish times of the nodes in this subtree are reduced by this factor of $(n-1)(k-1)s/B$. The finish time of the last child child remain unaffected by this and hence, the nodes in its subtree remain unaffected.

In this technique, we are delaying the beginning of transmission to the children that come later in the TDS order to finish transmission of data to the children earlier in the TDS order much sooner than otherwise. This observation shows that if the subtrees of node r are all of equal size, the nodes in child 1's subtree would finish much earlier than nodes in child k's subtree. This can be mitigated if the subtrees of the child nodes were distributed unequally, i.e., if the subtree of the first child were larger than that of the kth child. We build on this intuition in Section 4 where we propose tree construction algorithms that take TDS into account.

3 Analysis of TDS

Until now we have been considering the effect of using TDS to deliver data from a node to its children. We now develop a relation to calculate the finish time of an arbitrary packet in a data stream at any node in a TDS tree. For our model, we assume that the data to be transfered is composed of packets of size s. Several packets are aggregated to form blocks. We denote the number of packets in a block by n. Let m be the packet index in a stream of data that is being transfered to the clients.

The end hosts that are the targeted by application-layer multicast trees have diverse access links connecting them to the Internet [14], varying from dial-up links and cable and DSL modems to Ethernet. A characteristic of most of these types of access links is that they offer much larger download bandwidth to the node than upload bandwidth from the node (a factor of 10 with some ISPs using cable modems). Let the upload bandwidth available to each node a participating in the application-layer multicast tree be denoted by B_a. The bandwidth requirements of the application creates an upper bound on the number of children that a node can support. This upper bound can be given in terms of the number of children that a node a can support in an application-layer multicast tree and is given by $d(a) = \lfloor B_a/b \rfloor$. We make the assumption that in the tree, nodes with higher upload bandwidth are closer to the source, specifically for a node a with k children $c_0, c_1, \ldots, c_{k-1}$, $B_a \geq max(B_{c_0}, \ldots, B_{c_{k-1}})$. This assumption is reasonable as it has been shown in [7] that to obtain short trees with minimum propagation delay, the nodes with largest degrees should be closest to the source. Given our interest in improving the average latency to transfer data to the clients, any tree considered would have this property.

We begin by considering the simple case with a single source node r with k children. If $0 \leq m < n$ and node i is the j^{th} child of the source, the finish time $t_{i,m}$ of packet m at node i can be written as $t_{i,m} = nj(s/B_r) + (m+1)(s/B_r)$ where the first term represents the time to send the block to the j children before i and the second term represents the time to send the m packets to node i. In general, for $m > 0$, the finish time a packet can be broken up into three parts,

Table 1. Summary of Notation used

Symbol	Definition
B_a	Upload bandwidth of node a
n	Number of packets in a block
s	Size of a packet in bits
b	Bandwidth required by data stream
m	Packet index in a stream of packets
i	Index of node in a global numbering scheme
$p(i)$	Parent index of node i
$c(i)$	Number of child nodes of node i
$I(i)$	Index of node i in its parent's TDS schedule
$d(i)$	Maximum degree bound of node i

– time to transmit the packets of all previous blocks,
– time to transmit current block to the $(j-1)$ children preceding i,
– time required to transmit the packets of the current block to child i.

This can be represented by $t_{i,m} = (m/n)k(s/B_r) + nj(s/B_r) + (m+1)(s/B_r)$.

For a general application-layer multicast tree, we begin by making the observation that once the first packet of a block arrives at a node, the subsequent packets of that block arrive back-to-back. From our previous assumption about the upload bandwidth of a node being greater than or equal to that of its children, we know that the time to transfer a block to a node is less than or equal to the time that node takes to transfer the block to a child. In other words, once a node begins receiving a block from its parent, it can retransmit the block to its child without waiting for a packet to arrive. Therefore, the time a packet arrives at a node depends on the packet's arrival at the node's parent. We assume that once a node receives the first packet of a block, it immediately begins transmitting the packet to its children. We ignore the propagation delay of links in this analysis to make the exposition clearer but incorporating the delays is straightforward.

Let η represent the block index, i.e., the integer value m/n and let η_1 be the first packet of block η. Let the function $p(i)$ denote the parent of node i and the function $I(i)$, the index of i in its parent's TDS schedule. The time of arrival of a packet m at a node i can be computed as follows:

$$t_{i,m} = t_{p(i),\eta_1} + I(i)n(s/B_{p(i)}) + (m \bmod n + 1)(s/B_{p(i)}).$$

From the above relations, we note that the latency to any node is dependent on the size of the packet and the number of packets in a block. We evaluate the effect of these factors in Section 5.

4 Tree Construction

We now consider the problem of constructing trees that take into account TDS. Current algorithms for constructing application-layer multicast trees optimize for delay or bandwidth without considering the transmission delay at interior nodes. We wish to create trees that not only take into account the transmission delay but also optimize

```
Tree T = createTree(Nodes $N$, Source $s$, DegreeConstraints $d$,
                    Blocksize $B$)
  Sort the nodes of N in non-increasing order of degree
    constraints into $n_1$,$n_2$,...,$n_{|N|}$.
  T = {$s$}
  Compute $t(s)$ as the time to transmit $B$ to first available
    child of $s$.
  Insert $s$ into MinHeap with value $t(s)$.
  $i = 1$
  while $i \leq |N|$
    Get next node $a$ from MinHeap.
    Attach $n_i$ as child of $a$.
    T = T $\bigcup$ {$n_i$}
    if $c(a) < d(a)$
      Recompute $t(a)$ and insert $a$ into MinHeap with
        value $t(a)$.
    if $d(n_i) > 0$
      Compute $t(n_i)$ and insert $n_i$ into MinHeap with
        value $t(n_i)$.
    $i++$
  done
  return T
```

Fig. 3. Tree Construction Algorithm for TDS ignoring propagation delays

for the block size being used in TDS. We begin by stating the objective for our tree construction algorithm.

The optimization problem can be stated as follows: Let $G = (r, N, E)$ be a complete graph with a source node r and end hosts N. Let the degree constraints of the nodes be given by the function d. Let E be the set of edges between the nodes. Our objective is to find the tree T with minimum average finish time to transfer the block B and satisfies the degree constraints. We first consider the case where the end-to-end delay between any pair of nodes to be the same (in this case zero), i.e., we ignore the propagation delay but not the transmission delay caused by the link scheduling. We present a centralized algorithm in Figure 3 that constructs an optimal tree for this case. The algorithm is run by a designated node such as the source in the following manner: First, the nodes are sorted in non-increasing order of their degree constraints. In the main loop, the next node from the sorted list is selected and attached to the tree at the position with the minimum finish time until all nodes are attached. If no attachment points exist due to the degree constraints of the nodes, the tree returned is empty.

Theorem 1. *The algorithm createTree generates a tree such that the nodes have the minimum finish times given the degree constraints d.*

The proof relies on the following lemmas.

Lemma 1. *For any node in the optimal tree, the size of its childrens' subtrees are in the order in which data is sent to the children, i.e., if data is sent to child i before child j, the size of i's subtree is greater than or equal to j's subtree.*

Lemma 2. *For any node in the optimal tree, the child with the larger degree bound is sent data before a child with a lesser degree bound.*

A corollary to lemma 2 is that for any node, the child with the largest degree bound is the first to which data is sent.

Lemma 3. *Given a set of N nodes with degree constraints, the optimal tree has the node with the maximum degree constraint as the root.*

The sketch of the proof of lemma 3 is as follows: We assume, for contradiction, that the optimal tree does not have the node with the maximum degree constraint at the root. It follows that for some subtree in the optimal tree, the node a with the maximum degree constraint is the child of a node with a smaller degree constraint. By lemma 2, a is the first child to be sent data by its parent. We show that exchanging a with its parent and rearranging the subtrees of a and the subtrees of the parent such that the larger subtrees are attached to a leads to a tree with a lower finish time, violating our assumption that this tree was optimal.

The proof of lemma 2 is very similar. The complete proofs can be found in [13].

Proof (Theorem 1). We prove this by induction on i, the number of nodes attached to the tree. If $i = 1$, then there is only one node in the tree, the source s and it is clearly optimal. Assume that the algorithm creates an optimal tree for i nodes. By the algorithm, the degree constraint of the $i + 1$st node is less than or equal to the degree constraint of any node in the tree uptil now. By lemma 3, node $i + 1$ can only be attached as a leaf, and the position that minimizes the finish time of $i+1$ is the position with the minimum transmission delay from the source to $i + 1$ which is node a from the algorithm. Thus, the tree with $i + 1$ nodes is also optimal. □

The general problem of constructing optimal trees with non-uniform propagation delays between nodes has been shown to be NP-Hard in [2]. To handle tree construction for TDS taking propagation delays into account, we propose the following *TDS heuristic* that attempts to balance the degree bound of a node and its propagation delay to the tree. The heuristic iteratively adds nodes to the tree in the following manner: Initially, three sets of nodes are created, N_A consisting of nodes that are attached to the tree, N_{Av} consisting of nodes that are attached and can accept more children and N_U consisting of nodes that are unattached. Let $l(u)$ be the latency of node u to the source along the tree. In the beginning, N_A and N_{Av} contain only the source node and N_U contains all other nodes. At each iteration, the algorithm computes a cost for each $v \in N_U$ as

$$Cost(v) = \min_{u \in N_{Av}} \alpha l(u)/l_{max} + \beta d(v)/d_{max} + (1-\beta)\delta(u,v)/\delta_{max}$$

where $l_{max} = max_{u \in N_{Av}} l(u)$, $d_{max} = max_{u \in N_U} d(u)$ and $\delta_{max} = max_{u \in N_U, w \in N_{Av}} \delta(u,w)$ are normalization constants.

The variables α and β control the weight of the different factors in the computing the cost of each unattached node. The value of α controls the extent to which the latency in the tree to the attachment point affects the cost, while β determines the relative importance of the degree constraints and the propagation delay of the unattached nodes.

From our evaluation, we observed that the heuristic is relatively unaffected by the value of α and so we fixed the value of α to be 1. We explore the effect of the β parameter on the average latency in the next section.

5 Evaluation

Methodology. We used libraries provided by the *p-sim* simulator [8] to write our simulation. For our simulations, we begin by creating a representative Internet topology using GT-ITM [4] comprising of 4050 nodes. We then randomly choose some of the nodes to be the hosts participating in the application-layer multicast tree. We randomly select one of the nodes to be the source of the end-system multicast. The degree constraints for the nodes are assigned from a uniform distribution with the source node being assigned the maximum degree constraint. The link latencies are drawn from uniform distributions with [50ms, 200ms] for the transit links, [25ms, 100ms] for the transit-stub links and [5ms, 50ms] for the stub-stub links. The stream bandwidth requirements are set at 8kBps per child in the multicast tree. We construct the application-layer TDS multicast tree using the algorithm detailed above. For all the experiments we report the average finish time as the time to transfer 50 kB of data from the source to all the nodes in the tree. We chose block sizes of 50kB and 5kB as reasonable bounds on the size of an application's data unit. The packet size used is usually 500 bytes. We also experimented with 1500 byte packets but the results were similar with the 1500 byte packets having a slightly larger average finish time. We begin by investigating the different parameters that affect the TDS scheme.

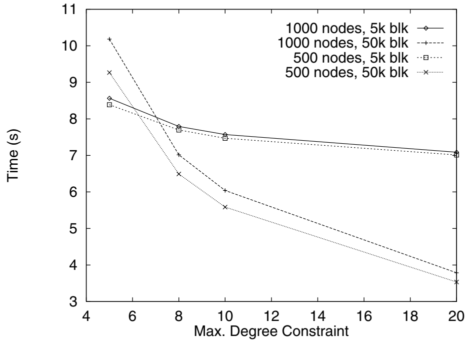


Fig. 4. Varying the maximum degree constraint of the nodes participating in the multicast tree

Effect of TDS Parameters. In Figure 4, we plot the average[3] finish time of the nodes in the TDS tree on the y-axis against the maximum degree constraints allowed for the nodes on the x-axis. Each line represents different size trees with varying block sizes. From the graph, we see that for smaller degree constraints, the smaller block sizes are

[3] In all cases, the median was less than the average. We omit plotting the medians for clarity.

Table 2. Node distribution of TDS trees for different block sizes

Max. Degree Constraint	Block size (kB)	Percentage of nodes in first two subtrees	Depth
10	50	66	8
	5	39	5
8	50	66	8
	5	39	6
5	50	75	9
	5	67	7

better for TDS. The small degree constraint results in trees that are tall and narrow, resulting in poor performance of TDS as the difference between the finish times of the first and last child at a node are not significant enough to offset the longer transmission delays. As the maximum degree constraint is increased, the trees created are wider and the larger block size has significantly better performance. The trees for the larger block size are more unbalanced with the subtrees of the children that are earlier in the TDS order being much larger that the subtrees of those later in the TDS order. This can be seen in the table in Table 2 in which we show the size of the subtrees in terms of the percentage of the total nodes that the subtree contains. Although the degree bounds for the nodes are assigned from a uniform distribution, the distribution of the degrees of nodes in the final tree is similar to the distribution of degrees observed by Sripanidkulchai et al [14].

Effect of β Parameter on the TDS Heuristic. In Figure 5, we plot the average finish time of the nodes in the TDS tree on the y-axis against various values of β on the x-axis. Each line represents trees constructed with a particular maximum degree constraint and each point is the average of five runs of the simulator with different seeds. We observe that the average finish time is only marginally affected for values of β up to 0.5. Actually the average finish time is reducing in this interval, but as the value of β increases above 0.5, the average finish time increases quickly. This is also seen in Figure 6, in which the curves are plotted for a block size of 5kB. These plots show that selecting the nodes primarily on the basis of the propagation delay to construct TDS trees results in poor performance. The best balance seems to be to equally weight the degree constraints of the nodes and their propagation delay when considering the next node to add to the tree.

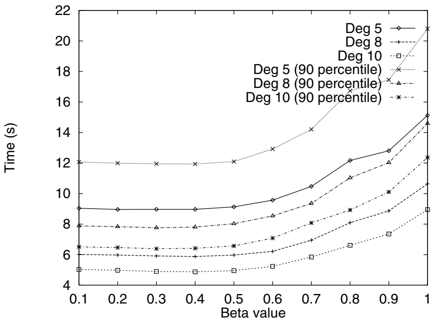


Fig. 5. Varying β with block size of 50kB

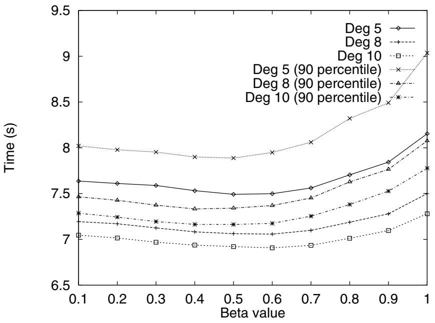


Fig. 6. Varying β with block size of 5kB

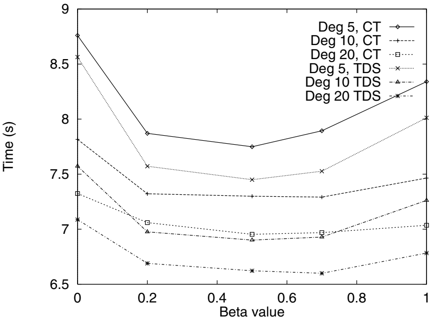


Fig. 7. Varying β with block size of 5kB using CT and TDS heuristics for 1000 nodes

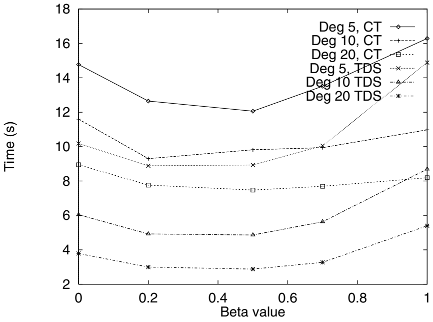


Fig. 8. Varying β with block size of 50kB using CT and TDS heuristics for 1000 nodes

We also plot the 90^{th} percentile value of node finish times for each of the degree constraints. We observe that the 90^{th} percentile value of the 50kB block with a degree constraint of 10 has a smaller finish time than the average finish time using 5kB blocks. This indicates that most of the nodes benefit when we use larger blocks. Another observation we can make from the graph is that the 90^{th} percentile value is closer to the average finish time for the 5kB block size than for the 50kB block size, indicating the increased variance due to the larger block size.

Planet-Lab Experiments. To evaluate the effects of TDS in a real-world scenario, we developed a small application to run on the PlanetLab network [11]. In a limited experiment using 16 nodes, block sizes of 50kB and 5kB, a packet size of 1000 bytes, we observed that the average finish time of the 50kB block size was 6.32 seconds while for the 5kB block was 8.41 seconds which agrees well with our analysis.

Performance Relative to Existing Heuristics. There have been other heuristics proposed to construct degree-bounded trees for application-layer multicast. The heuristics proposed are for minimizing the maximum latency to clients [12] (which we call Compact Tree) and for minimizing the cost of using proxies [7] (which we call Min Cost). The Compact Tree heuristic incrementally constructs a minimum spanning tree from the source s. For each node v not in the tree, it finds the minimum cost edge (u, v) from a node u in the MST. The cost that is minimized is the overlay delay $\delta(s, v)$ from the source to the node v. The Min Cost heuristic is quite similar except for the cost function used to select the next node. The Min Cost heuristic considers the minimum latency edge (u, v) as well as the degree constraint d of the nodes while selecting the best node to attach to the tree. The cost function is $\gamma_\alpha(v) = \alpha d(v)/d_{max} + (1-\alpha)\delta_{min}/\delta(s, v)$. The α parameter plays the same role as the β parameter in the TDS heuristic and d_{max} and δ_{min} are normalization constants. Both heuristics do not consider the cost of transmission of data while constructing the trees.

For our simulations, we implement both heuristics using the same routine. The value of the parameter $\beta = 0$ creates trees based on the Compact Tree heuristic while other values of β create trees based on the Min Cost heuristic. We plot the average finish times for delivering 50kB of data using trees with 1000 nodes constructed by the different heuristics in figures 7 and 8 for block sizes of 5kB and 50kB respectively. In general, the graphs show that the TDS heuristic performs much better for every degree bound that is used as it considers the transmission delays incurred at each node. The magnitude of the improvement of the TDS heuristic is larger with block sizes of 50kB than with 5kB. The trees created by the TDS heuristic for the 5kB blocks are not very different from the trees created by the other heuristics and so the improvement seen is on the order of a second in the average finish times. On the other hand, the trees for the 50kB block size created by the TDS heuristic exploit the larger block size to create trees that are very different from those created by the other heuristics resulting in significant improvement over the other heuristics. When β is in the range of 0.5 to 0.7, the Min Cost and Compact Tree heuristics perform best as they consider a combination of the degree constraints along with the propagation delay to the nodes in the tree.

6 Related Work

In [12], the authors describe the problem of creating minimum diameter degree bounded spanning trees and show that it is NP-Hard. They propose a greedy heuristic to create trees based on this objective. In [7], the authors define the cost of a tree as the number of special proxy nodes used to create multicast trees. Using this they propose to create trees which satisfy a maximum delay bound while minimizing cost. They provide an optimal solution for graphs with uniform edges and show that this problem is NP-Hard in the general case with non-uniform edges.

In [2], the minimum average-latency degree-bounded directed spanning tree problem is introduced in context of a two-tier infrastructure for implementing large-scale media-streaming applications. The infrastructure, called OMNI (Overlay Multicast Network Infrastructure) consists of a set of Multicast Service Nodes (MSNs) to which end-hosts connect to form the multicast tree. The objective of this work is to reduce the average latency to the end-hosts. This is achieved by arranging the MSNs to create minimum latency trees where each MSN is weighted by the number of clients connected to it. The authors impose a degree bound on each MSN but do not account for the transmission delays at the MSNs which we consider in this work. Also, we focus on application-layer multicast trees without any explicit infrastructure in the network.

In [3], the authors point out that the models used currently to construct these trees neglect to consider the fact that the most nodes in an end-system multicast tree have a single network connection and this connection has to be shared between all the children of the node. The authors propose an overlay network model to account for these costs and propose heuristic algorithms to construct multicast trees that consider the transmission and computation delays at each of the nodes in the multicast tree. The overlay model proposed does not explicitly consider the degree constraints at nodes. The construction of the tree is based on minimizing the delay to hosts but does not consider the effect of the degree constraints imposed by the access link bandwidth of the nodes or the effect of the access link scheduling on the average delay of the nodes in the tree.

7 Discussion and Future Work

Our results show that nodes sending large blocks of data to each child in turn can reduce the average finish time of nodes in the multicast tree. The tradeoff involved in this gain is the increased variance of the actual finish times of nodes. Based on this tradeoff, the block size and packet size for TDS can be specified to match application requirements. For example, the increased variance with larger block sizes can be used as an incentive mechanism to encourage nodes to dedicate more uplink bandwidth to the application. This in turn would place those nodes in positions where their finish times are earlier.

In this work, we examine the effect of the single access link that many end hosts that participate in application-layer multicast have. We show that the average finish times of nodes in the tree are affected by the way in which this link is used to transfer data to a node's children. We proposed a technique called Time Division Streaming to share this access link such that the average finish times are reduced as compared to previous work. We also provide analytical results based on a limited model of this technique and propose heuristics that take this serialization into account when constructing the tree.

Using the TDS heuristic to construct multicast trees, we show significant reduction in the average finish times of nodes. The heuristic exploits the effect of TDS by creating trees such that the interior nodes have unequal subtrees with the subtrees of children earlier in the TDS schedule being larger. In future work, we plan to create distributed version of the TDS heuristic that can be used to add nodes to trees as they arrive. Another direction of work is to vary the TDS parameters to tailor them for specific applications and to study the effect of these customizations.

References

1. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *SIGCOMM*, Aug 2002.
2. S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Infocom*, April 2003.
3. E. Brosh and Y. Shavitt. Approximation and heuristic algorithms for minimum-delay application layer multicast trees. In *Infocom*, March 2004.
4. K. Calvert, E. Zegura, and S. Bhattacharjee. How to model an internetwork. In *Infocomm*, 1996.
5. M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *SOSP*, Oct 2003.
6. Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. In *IEEE Journal on Selected Areas in Communication*, Aug 2001.
7. N. M. Malouch, Z. Liu, D. Rubenstein, and S. Sahu. A graph theoretic approach to bounding delay in proxy-assisted, end-system multicast. In *IWQoS*, May 2002.
8. S. Merugu, S. Srinivasan, and E. Zegura. p-sim: A simulator for peer-to-peer networks. In *MASCOTS*, October 2003.
9. V. Padmanabhan, H. Wang, and P. Chou. Resilient peer-to-peer streaming. In *ICNP*, 2003.

10. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. Almi: An application level multicast infrastructure. In *USITS*, Mar 2001.
11. PlanetLab. http://www.planet-lab.org/, 2004.
12. S. Shi, J. Turner, and M. Waldvogel. Dimensioning server access bandwidth and multicast routing in overlay network. In *NOSSDAV*, Jun 2001.
13. S. Srinivasan and E. Zegura. Scheduling uplink bandwidth in application-layer multicast trees. Technical Report GIT-CC-04-14, Georgia Tech, 2004.
14. K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *SIGCOMM*, Aug 2004.

A Novel Packet Marking Function for Real-Time Interactive MPEG-4 Video Applications in a Differentiated Services Network

Shane O'Neill, Alan Marshall, and Roger Woods

The Institute of Electronics, Communications and Information Technology (ECIT),
Queen's University of Belfast,
Queen's Road, Belfast, BT3 9DT, UK
{shane.oneill, a.marshall, r.woods}@ee.qub.ac.uk
http://www.ee.qub.ac.uk/dsp/

Abstract. The future convergence of voice, video and data applications on the Internet requires that next generation technology provides bandwidth and delay guarantees. Current technology trends are moving towards scalable aggregate-based systems where applications are grouped together and guarantees are provided at the aggregate level only. This solution alone is not enough for interactive video applications with sub-second delay bounds. This paper introduces a novel packet marking scheme that controls the end-to-end delay of an individual flow as it traverses a network enabled to supply aggregate-granularity Quality of Service (QoS). IPv6 Hop-by-Hop extension header fields are used to track the packet delay encountered at each network node and autonomous decisions are made on the best queuing strategy to employ. The results of network simulations are presented and it is shown that when the proposed mechanism is employed the requested delay bound is met with a 20% reduction in resource reservation and no packet loss in the network.

1 Introduction

Today, multimedia applications like digital television services and video conferencing require dedicated networks, or expensive leased lines in circuit-switched networks, to provide the required level of quality. In the future these distributed applications will be connected over packet-based IP networks as more carriers move towards all-IP technology. For example, British Telecom hopes to completely replace its current UK circuit-switched network with an all-IP network within 5 years. To enable this transition, IP networks which traditionally provided a "Best Effort" service must be enhanced to provide levels of QoS necessary for transporting multimedia applications. This has led to the development of the Differentiated Services (DiffServ) architecture [1]. DiffServ groups traffic together based on its classification, such as aggregating all real-time traffic as a single class. A traffic class can then be guaranteed a minimum level of service. This mechanism will allow the isolation of multimedia traffic from traditional best effort traffic on a common network. Although this traffic segregation

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1031 – 1042, 2005.

may provide the required level of service for delay-insensitive applications, it is not sufficient for services such as video conferencing and video telephony. The heterogeneous nature of the constituent flows of a real-time aggregate and the variation in end-to-end flow path lengths requires the aggregate to be bound to the delay of the worst case flow, which is an inefficient use of network resource. Enabling these real-time services is a necessity if the transformation to all-IP networks is to be realised.

This paper details the analysis of interactive compressed MPEG-4 video flows transported as an aggregate on a DiffServ enabled network. It is shown that whilst aggregate reservations can provide the long-term average bandwidth required by the individual video applications, the self-similar nature of an aggregate of video flows requires an inefficient link reservation that is well above the mean aggregate bandwidth in order to control end-to-end delay. This paper introduces a novel packet marking scheme that utilizes IPv6 hop-by-hop extension header fields and requires no per-flow state information to be stored at internal network nodes. This novel scheme can control end-to-end delay with a reservation rate close to the average requirement. To the authors' knowledge this mechanism has not been previously presented in the literature.

The paper is structured as follows. Section 2 describes the traffic demands of a compressed video application and current state of the art QoS network techniques in more detail. The novel packet marking scheme is introduced in Section 3 and simulation results from OPNET Modeller are presented in Section 4. Finally, Section 5 provides a brief summary and future direction of the research.

2 Background

2.1 Video Compression and Video Application Traffic Demands

Video traffic has a relatively high data-rate and in the future this will lead to much higher network loads as the use of video applications such as VoD and video telephony become prolific. One method of reducing the load on network resources is to employ compression on the video source before it is transmitted. MPEG-4 visual [2,3] is a recent video compression standard that is being used by consumers. The transport requirements of a video stream are dependant on the end-to-end delay constraint and whether or not the source is pre-encoded. If the source is pre-encoded multiple passes of the encoder can produce more efficient compression and an online video scheduler can be used to pre-fetch future frame data to smooth the output data-rate [4,5]. Depending on the tolerable delay, smoothing at the source can be applied to reduce peaks in the output data rate with a playback buffer being used to absorb jitter. Table 1 shows three types of streaming video applications and their different network constraints. It can be seen that interactive video applications like video conferencing are completely dependant on the intermediate network technology to provide a high level of quality to the end user.

Table 1. The Transmission Constraints for Common Video Applications

Application	Encoding	Pre-fetching	Smoothing	Delay
Digital Television	Pre-encoded	Yes	Yes	Seconds
Live Broadcast	Realtime	No	Yes	Seconds
Video Conference	Realtime	No	No	Sub-seconds

2.2 Next Generation Network Architectures

The introduction of new multimedia applications with bandwidth and delay constraints creates a requirement for new traffic processing mechanisms to handle them. The DiffServ architecture [1] provides service differentiation between different traffic classes and QoS guarantees. This allows different classes of traffic to receive different levels of service. DiffServ operates on traffic aggregates in the core of a network. This provides a scalable solution that minimises the amount of state information stored at each core network node. All flows with similar service requirements are processed together as an aggregate. The edge node, or possibly the customer, chooses the service level by setting the DiffServ Code Point (DSCP) in each packet header. From this point forward, unless reclassification occurs, the flow will be queued with other flows with the same DSCP mapping and experience the same service as them. A number of standardised mappings already exist including Expedited Forwarding (EF) [6] and Assured Forwarding (AF) [7]. To provide different levels of service to each class, it is necessary to schedule packets. The purpose of the packet scheduler is to divide the link capacity fairly between competing classes at the desired ratio. Fig. 1 shows a common configuration for a DiffServ queuing mechanism at the output port of a network switch.

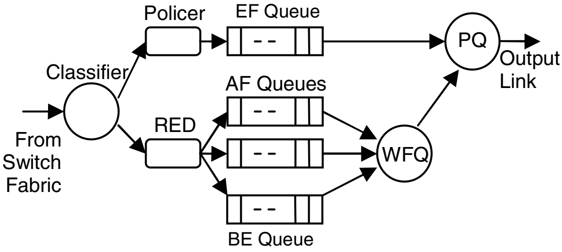


Fig. 1. Output Port Queuing Mechanism for DiffServ

The Priority Queue (PQ) scheduler will always serve the EF queue if a packet is present. The input of the EF queue is policed to limit the volume of traffic using this queue. If the EF queue is empty, the Weighted Fair Queue (WFQ) scheduler services the next queue with the earliest timestamp. At minimum, the WFQ scheduler can service queues at their requested service rate. Random Early Discard (RED) [8] is used at the input of each queue to monitor queue levels and drop traffic dependant on the resultant probability graph. In this scenario, RED uses multiple parameter sets for the different drop precedence's of an AF Per-Hop Behaviour Group (PHB-G).

There has been a great deal of research investigating the benefits of using DiffServ to provide service guarantees to real-time and non real-time traffic [1,6-10]. A few in particular have focused on dynamically modifying the DiffServ classifications or queuing configuration within the network to respond to varying network conditions [11,12]. The work proposed in this paper differs in that it uses knowledge of the profile of aggregated video traffic along with hop-by-hop headers to control the end-to-end delay without dynamically changing the resource reservation of the traffic classes.

3 A Novel Packet Marking and Queue Management Mechanism

Frame lengths of MPEG encoded video traffic vary in size and are said to be self-similar, with frame size distributions exhibiting long-range dependence. This results in the presence of bursts at multiple timescales. Due to the self-similar nature of the traffic profile, handling bursts of all sizes, without exceeding the delay tolerance, requires a reservation far in excess of the mean. This is costly and inefficient as most of the time the source does not use the bandwidth reserved. It is important to note that an aggregate that contains self-similar traffic flows is itself self-similar, but there is still some multiplexing gain to be achieved, as described in [15].

Before any delay-based packet marking schemes can be devised it is important to understand the delay experienced by a packet traversing a DiffServ-enabled network. First, if a single-flow scenario is considered, the arrival rate of a video stream to the first network node conforms to a long-term average data rate but can vary over short-term timescales. If the first node is configured to service the queued video stream at the long-term average data rate, then under loaded traffic conditions the output stream will be shaped and packets will be output at this data rate. Any bursts arriving above the service rate will be queued. If the remaining nodes in the path of the traffic flow are configured to the same service rate, then packets will depart these nodes as quickly as they arrive, and in the case where all video frames are fragmented to packets of a similar size, queue levels at the remaining nodes will be no more than one packet. This analysis leads to a worst-case end-to-end delay bound characterised by equation (1).

$$D \le \underbrace{\frac{BT}{r}}_{A} + \underbrace{\frac{(n-1)*F}{r}}_{B} . \tag{1}$$

In (1), *D* is the maximum end-to-end delay in seconds, *BT* represents the burst tolerance at the first hop which is the maximum queue size an arriving packet can tolerate, *F* is the fragment size and *n* is the number of hops from host to host, and r is service rate in bit/s, which is the same at each hop. *BT* and *F* are measured in bits.

The delay is worst-case in that it assumes that a flow only gets its assured service rate at each node. Equation (1) has been validated in simulations to accurately characterise the end-to-end delay for a packet in a single-flow scenario. Part A is the delay experienced at the first hop, while Part B is the delay experienced by a fragment as it traverses the remaining hops. Under aggregate scenarios, however, it can be shown that Equation (1) needs to be modified. The aggregate service rate is assumed to be the accumulated individual average rates. The queuing delay at the first hop is still effectively the same as the single flow scenario, but, the portion of delay associated with the traversal of the last fragment across the network is now considerably smaller. The queue levels at the first hop are now an order of magnitude higher than the single flow scenario, so the first hop delay is similar to that of the single flow case. The first hop releases traffic into the network at an aggregate service rate and, in this scenario, subsequent nodes are configured at the same aggregate rate. Therefore, the arrival rate and departure rate at these hops are equal so queue levels are the same as the single flow case and consequently part B of the delay in Equation (1) is much lower.

Using this delay analysis a novel packet marking scheme has been devised to control the end-to-end delay of a compressed video packet through a DiffServ enabled network by employing two delay fields within a packet header. The IPv6 hop-by-hop options extension header can provide a suitable position for these fields. The end-to-end delay is characterised by a “first hop” delay and an “other hop” delay, as described in the paragraph above. These two fields are added to the packet by the video source and are used within the network to determine if the current network node’s queue level will cause the packet to be delayed beyond the respective delay threshold. The video traffic aggregate uses an AF queuing mechanism that provides a bandwidth guarantee through a WFQ scheduler. At each hop the queue level that an incoming packet can tolerate is calculated based on the delay allowed at that hop. To estimate the two delay values the only network parameter needed is the approximate number of hops in the path, as shown in Equation (1). The burst tolerance at the first hop, *BT*, can be estimated at the source by monitoring the output traffic profile. Equation (2) shows the inequality used for monitoring queue levels.

$$\mathrm{T_H} * r_q \geq Q + L \,. \tag{2}$$

In (2), T_H represents the hop delay tolerance in seconds, which is either the “first hop” field or the “other hop” field depending on the current network node, Q denotes the current queue size in bits, L is the incoming packet size in bits and r_q is the service rate of the queue in bit/s. If adding the incoming packet L to the specified AF queue does not cause the queue level to exceed T_H* r_q, the maximum number of bits serviced within the hop delay tolerance, the packet is admitted to that AF queue. If the AF queue level is exceeded, the packet is requeued at an EF queue. Traditional shaping of individual flows at the ingress to the network does not make efficient use of the aggregate bandwidth. This mechanism provides a simple solution to constraining the delay of a single flow that is processed within an aggregate without shaping individual flows at the network ingress during bursty periods.

If a traffic profile is self-similar it can be approximated by a heavy-tailed distribution similar to the one depicted in Fig. 2 [14].

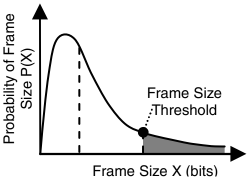


Fig. 2. Frame Size Distribution (Lognormal)

This shows that large frames are infrequent events. If we consider only the tail of the distribution, this would produce a very low bandwidth, high burst stream. The approach presented here is that an EF queue, serviced by a PQ scheduling mechanism, is used to transport the infrequently large bursts across the network. This prevents a persistent increase in delay at the first hop and swiftly passes the burst across the network. To constrain usage of the overflow path a policer is employed at the ingress to the EF queue, as shown in Fig. 1. The policer can be set with a low token rate and high burst tolerance. This permits bursts, but not very often, which should match the profile of the end of the tail. Effectively, the heavy tail of the distribution has been removed, as indicated in Fig. 2 by the shaded region.

To summarise, the packet marking mechanism presented utilizes two fields in the packet header to indicate “first hop” and “other hop” delays and requires intermediate nodes to process these fields and determine whether the current queue levels would cause intolerable delay to the incoming packet. Any packet that would experience excess delay at a node is redirected to the EF queue. The procedure used to calculate the delay fields, the simulation environment, and the results of the simulations are described in the following section.

4 Simulation Environment and Results

4.1 Simulation Topology and Setup

The network topology used to test the proposed delay mechanism is shown in Fig. 3. The chosen topology represents a realistic network scenario with traffic originating from individual LANs but then merging with other traffic throughout the network. Two pairs of video flows from LAN 1 are forwarded to LAN 3 and 4. The same occurs for video flows from LAN 2. This merges into an aggregate of 8 flows in the backbone and splits into a 4 flow aggregate for both LANs 3 and 4, illustrated in the figure by the various dashed lines. An example of this scenario would be video traffic originating from different UK enterprises that is destined for either the US or mainland Europe. Depending on how many ISP domains exist between the source and destination, the size of the aggregate path may vary. The service rate at each output port is dependant on the number of flows present.

Two network components have been developed in OPNET Modeller to perform the required simulations. A video client that can generate and terminate MPEG compressed video traffic and a switch that implements the queuing mechanism as shown

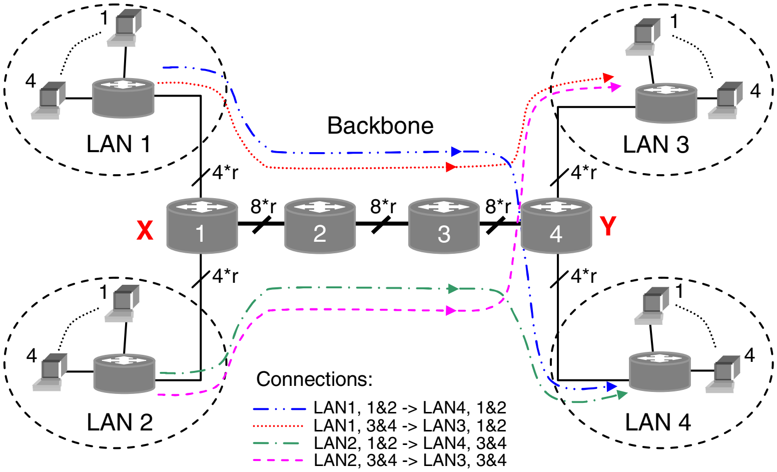


Fig. 3. The Network Topology for Simulation in OPNET Modeller

in Fig. 1. An analytical MPEG-2 model is used to generate the video frame sizes, which was developed for OPNET Modeller by Deshpande and Kandala of Sharp Laboratories [16]. This model will produce an output traffic profile similar to that of an MPEG-4 encoder that is set with the parameters for a video conference scenario, i.e. a single virtual object and single pass of the encoder. The video client originally produced I-frames, P-frames and B-frames that independently conformed to a log-normal distribution. The generator has been enhanced to allow configurable fragmentation of the video frame and now contains the required delay header field functionality. The video client also provides data-sink capabilities for terminating video frames and producing end-to-end statistics.

The network switch component performs the queuing mechanism, shown in Fig. 1, at each output port. The AF queues can be configured to have a desired share of the link bandwidth which is achieved by a WFQ scheduler. The WFQ scheduler is configured to use a non-working conserving mechanism to emulate a busy node. RED is not employed at this stage in the research as video traffic is being remarked, and not dropped, based on queue levels. The proposed delay threshold check is performed at the input to an AF queue. The input of the EF queue is policed to control the volume of traffic using EF resources.

The video clients, shown in Fig. 3, are set up to output a Group of Pictures, GOP (15,1), structure with a frame rate of 30 fps. This is a common GOP structure for NTSC (US television) quality video. This provides two I-frames per second, each one followed by 14 P-frames as shown in Fig. 4. B-frames cannot be used for interactive traffic as their generation incurs too much delay at the encoder.

The mean and variance for each distribution are taken from analysis of an MPEG-4 trace statistics file available on the Internet [17]. The parameters used are listed in Table 2. The values in Table 2 represent a 176x144 pixel resolution QCIF encoded MPEG-4 trace of a lecture theatre scenario which is similar in content to a video-conference.

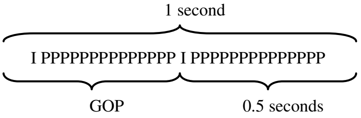


Fig. 4. The MPEG GOP Structure

Table 2. Lognormal Frame Size Parameters

Frame Type	Mean	Variance
I	30,000	34,500,000
P	8,000	16,000,000

The average I-frame and P-frame sizes of 30 kbit and 8 kbit respectively, give a long-term average bandwidth requirement of 284 kbit/s. The frames are fragmented into 500 byte packets as they leave the video clients. This fragment size has been chosen to provide a reasonable number of fragments for the range of frame sizes produced by the video client. As they traverse the network, all packets are queued at the same AF queue serviced by a WFQ scheduler. Each switch node is configured to service this queue so that it receives the desired rate. For the topology shown in Fig. 3, the queue service rate at the output port of each switch node is set to 320 kbit/s * *number of flows in aggregate*. The slightly higher rate of 320 kbit/s has been chosen to allow for small fluctuations in rates above the average which handles most arrival rates except in cases of excessive loads, i.e. the heavy tail of the frame size distribution.

4.2 Hop Delay Field Simulation Results

Using Equation (1) and assuming a base service rate of 320 kbit/s and a burst tolerance of 40 kbits at the first node, a "first hop" delay of 0.125 seconds is calculated. The burst tolerance of 40 kbits has been chosen from previous analysis of single flow scenarios.

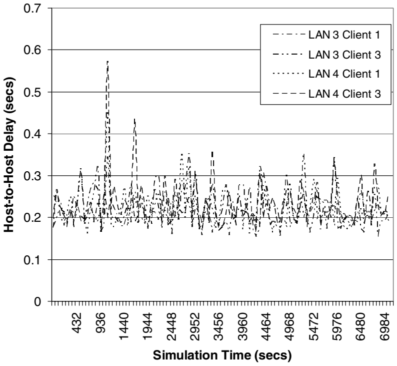


Fig. 5. End-to-end Delay With No Delay Threshold Checks

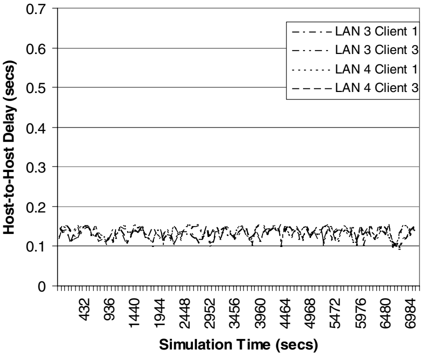


Fig. 6. End-to-end Delays with Delay Threshold Checks

The end-to-end delay tolerance is assumed to be 0.2s. Video conferencing applications have a delay tolerance of 0.3s from capture to display. Choosing a network delay of 0.2s allows 0.1s for encoding and decoding. The "other hop" delay field is therefore, (0.2 – 0.125)/(6-1) = 0.015s. The results of the simulations are illustrated in the graphs in Fig. 5 and Fig. 6.

The graphs show the Host-to-Delay for 4 of the flows in the network topology illustrated in Fig. 3. When no delay checks are performed in the network, the end-to-end delay often exceeds the threshold of 0.2s. When the delay checking mechanism is enabled the delay control mechanism performs its task and the resulting delays are below the 0.2s limit requested. However, further analysis reveals that the volume of traffic utilizing the expedited path is very high, as shown in Fig. 7. From single-flow analysis, the amount of overflow traffic utilising the expedited path was observed to be below 10% of the average throughput. In an aggregate scenario, statistical gain should reduce this percentage further. Fig. 7 shows that approximately 50% (of 284 kbit/s average) of the traffic is being switched to the EF class which is much higher than observed in the single flow scenario. With further analysis the source of the problem is found to be unstable queue levels at network nodes after the first node, as shown in Fig. 8, which represents the AF queue levels at switch 4 in Fig. 3.

The first node releases traffic at the aggregate rate and it was expected to produce stable, low queue levels at the remaining nodes and thus low queue delays. However, the merging of the two 4-flow aggregates at point X in Fig. 3 can lead to simultaneous arrivals from both ingress aggregates. The output aggregate rate is large enough to keep queue levels to a minimum. However, as observed over a short time interval, a burst of consecutive packets destined for the same LAN can cause an increase in queue levels at point Y, where the aggregates split. At point Y it is possible that over a short timescale, the arrival rate of traffic to the output port destined for either LAN 3 or LAN 4 is equal to the rate of the aggregate flow between X and Y. This causes the temporary build up in queue levels.

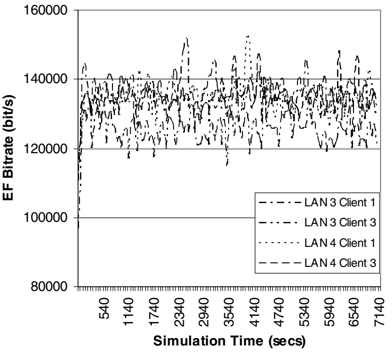


Fig. 7. Destination Host Received EF Bitrate

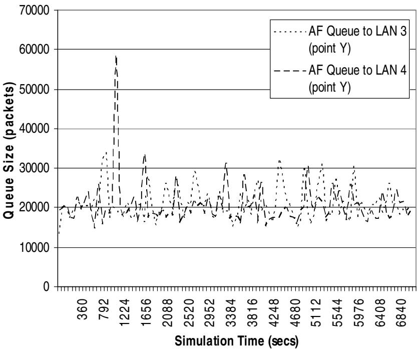


Fig. 8. Output Port AF Queue Sizes (Switch 4)

It would be very difficult to provide a solution to this problem at the merge point, i.e. the source of the problem, as any solution at this node would need to know the routes that flows traverse across the network. A better solution is to improve the intelligence of the delay threshold monitor at each AF queue.

When packets arrive at a node, depending on current queue levels, they may leave the node earlier than their delay tolerance. This provides these packets with a little bit of "slack" that could be used at other nodes with higher queue levels. Therefore, an additional mechanism is added at the output of the AF queue that introduces a "slack" packet field with any excess time gained by leaving the queue early.

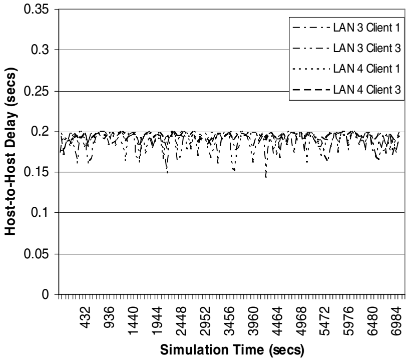


Fig. 9. End-to-end Delays with Delay Threshold Checks and Slack Adjustment

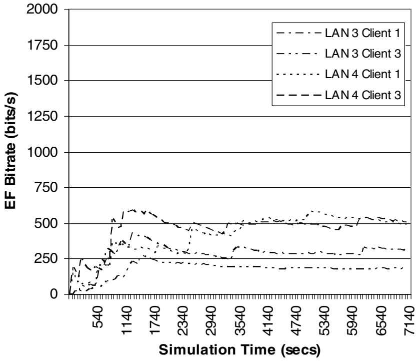


Fig. 10. Destination Client Received EF Bitrate with Slack Adjustment

Fig. 9 shows the end-to-end delay of a selection of flows from a simulation with the slack monitoring function present. At each switch node the hop delay threshold of the incoming packet, and now also any slack gained at previous nodes, is checked against current queue size. When the packet is transmitted, any slack gained at the current node is added to the "slack" field in the packet header. From Fig. 9 it can be seen that delay is now bounded tightly below 0.2s as requested, but more importantly, a comparison of the graphs in Fig. 7 and Fig. 10 shows the volume of traffic received at the destination video hosts has been reduced from approximately 130 kbit/s to less than 1 kbit/s.

Table 3. Simulation Results, Base Service Rate of 320 kbit/s

Scenario	Basic	Delay	Delay + Slack
% E2E Delay > 0.2s	3%	0	0
Expedited Data Rate	n/a	130 kbit/s	1 kbit/s

4.3 Comparison of Results

Utilising the three timing fields, a reservation of 320 kbit/s per flow in the assured path, with the EF policers set to 1 kbit/s token rate and 40 kbit burst tolerance, the end-to-end delay is effectively constrained below 0.2s for all flows. To test the limits of the presented mechanism, a 25% increase in aggregate service rates, with the mechanisms disabled, found violations of the 0.2s threshold still present.

5 Conclusions and Future Work

This paper proposes a novel mechanism that bounds the delay of a real-time interactive video application when transported across a network that only provides aggregate guarantees. Using three packet header fields and queue management at network nodes the end-to-end delay of VBR video traffic profile can be controlled. This paper proposes the use of an expedited path in the DiffServ network to handle AF queue overflows. The expedited path is policed to restrict the volume of traffic and acts as an express lane to quickly remove congestion from the AF queue. Using OPNET Modeller simulations show that, through the use of packet marking, the end-to-end delay of a video flow can be bounded below a desired level and a saving in resource reservation of at least 20% can be achieved.

In this investigation all individual flow rate requirements are the same and aggregate services rates have been just enough to meet delay bounds. Future work will investigate the effect of varying and larger reservation rates than required, which is a likely occurrence in a network already provisioned for aggregate traffic. The addition of background traffic to the simulations is also necessary to fully understand the effect of using the expedited path for excess video traffic. If a global solution can be found, it is hypothesised that the implementation of this may be suited to an FPGA-based programmable platform for sufficient wire-speed processing power and to allow customisation once in place in a real network. There is also a potential autonomous aspect to the solution, whereby the operation of the delay control mechanism is monitored to detect mis-configuration in a network, e.g. continual high usage of the expedited path would suggest the average AF queue input rate is higher than the reservation.

Acknowledgements

The authors acknowledge this work is supported and funded by Xilinx Inc. as part of the Pippin project. The authors would like to thank Gordon Brebner and Phil Roxby for their guidance and Wilson Chung for his assistance with the technical details of using MPEG-4 for compressed video in interactive applications.

References

1. S Blake et al, “An Architecture for Differentiated Service”, RFC 2475, Dec. 1998.
2. ISO/IEC 14496-2, “Information technology – coding of audio-visual objects – Part 2: Visual”, MPEG-4 Standards, First Edition, December 1999.
3. I.E.G. Richardson, “H.264 and MPEG-4 Video Compression”, Wiley, 2003.
4. C. Bewick et al, “Network Constrained Smoothing: A technique for Enhanced Network Support of Video Traffic”, *PGNET Symposium*, June 2001.

5. W. Feng, "Rate-Constrained Bandwidth Smoothing for Delivery of Stored Video", *SPIE Multimedia Networking and Computing 1997*, 1997.
6. B. Davie et al., "An Expedited Forwarding PHB", RFC 3246, Mar. 02.
7. J. Heinanen et al, "Assured Forwarding PHB Group", RFC 2597, June 1999.
8. S. Floyd & V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking, vol. 1, pp. 397-413*, Aug. 1993
9. I. Stoica & H. Zhang, "Providing Guaranteed Services without Per Flow Management", *Proceedings of SIGCOMM '99, pp. 81-94*, August 1999.
10. C. Dovrolis, D. Stiliadis & P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," *Proc.s of SIGCOMM '99, pp. 109-120*, Aug 99.
11. J. Shin et al, "Dynamic QoS Mapping Control for Streaming Video in Relative Service Differentiation Networks", *European Transactions on Telecommunications, vol. 12, no. 3, pp. 217-230,* June 2001.
12. T. Ahmed et al , "A Measurement-Based Approach for Dynamic QoS Adaptation in DiffServ Networks", *Journal of Computer Communications, Special issue on End-to-End Quality of Service Differentiation,* 2004.
13. M. Garrett, & W. Willinger, "Analysis, Modelling and Generation of Self-Similar VBR Video Traffic", *Proceedings of ACM SIGCOMM '94, pp. 269-280*, August 1994.
14. M. Krunz & H. Hughes, "A traffic model for MPEG-coded VBR streams", *Proceedings of the ACM SIGMETRICS '95 Conference, pp. 47-55,* 1995
15. B. Bashforth, & C. Williamson, "Statistical Multiplexing of Self-Similar Video Streams: Simulation Study and Performance Results", *Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 1998.
16. MPEG-2 model created for OPNET Modeller by Sachin Deshpande and Srinivas Kandala, Sharp Laboratories of America, Inc.
17. http://www-tkn.ee.tu-berlin.de/research/trace/pics/FrameStat/statb9f8.html.

On Multipath Routing with Transit Hubs

A. Sen[1], B. Hao[1], B.H. Shen[1], S. Murthy[1], and S. Ganguly[2]

[1] Dept. of Computer Science and Engineering,
Arizona State University, Tempe, AZ 85287-8809, USA
{asen, binhao, bao, sudhi}@asu.edu
[2] Dept. of Broadband and Mobile Networks,
NEC Laboratories, USA
samrat@nec-lab.com

Abstract. Empirical studies report frequent occurrences of path failure in the Internet. In providing resilience to such failures, we propose the computation of alternate backup end-to-end path that is disjoint to the default IP path. This disjoint path is created using transit hubs that can be located at diverse points in the Internet. Transit hubs provide better utilization of network resources. Assuming an IP layer routing between any two nodes, we show that the problem of computing such a disjoint path is NP-complete. We present an exact and a heuristic solution for the problem. Using routing data obtained from PlanetLab, we evaluate the efficacy of our heuristic solution.

Keywords: transit hub, disjoint path, multipath routing, network resilience.

1 Introduction

Current Internet routers select only a single path between a source and destination node. The choice of this default IP path is not left to the end hosts, instead it is left to the Administrative (AS) domain operators or on the BGP level policies. Often it is desirable for the end hosts to have better control over the route selected in context of traffic engineering and QoS controlled applications. A solution framework that has been gaining immense interest recently in the research community is using transit hubs. Transit hub routing allows routing between end hosts via a set of dedicated transit nodes that are placed at diverse locations on the Internet.

The benefits of transit hub routing are multidimensional. Transit hubs can forward packets, compute an end-to-end alternate path by chaining a set of default IP paths and thus facilitate better utilization of network resources. It provides better control over the load distribution in a network and can route packets over less congested areas. An interesting application area is bulk data transfers. Transit caches were deployed on the Internet in experiments [7], which resulted in transmission speeds of up to 47.6 Mbps during transfers of 3 TB of data between SanDiego and Urbana-Champaign. Bulk data transfer is not

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1043–1055, 2005.

readily supported by current Internet routers and would have been improbable without transit hubs. Another ongoing research effort in this direction is the Logistical Networking project [2] in which the transit hubs (called depots) offer middleware-like storage services that can be used in many applications.

Relying just on the single default IP path may lead to various end-to-end performance bottlenecks. Experimental studies conducted by authors of *detour* indicate that in many cases, alternate paths have better latency and throughput characteristics than direct default IP paths. Albeit the strong case for alternate paths, they cannot be directly constructed using existing routers as they do not provide any rerouting flexibilities. Transit hubs provide an elegant solution framework (using techniques like IP-in-IP encapsulation [4], overlay networks [1] or flexible extension headers of IPv6 datagrams) for creating alternate paths between end hosts and works seamlessly with the existing routers.

There is immense literature on traditional multipath routing problem[3, 10, 8] (others not provided due to space constraints). The advantages of multipath routing can be exploited to its fullest extent if the paths are link (or node) disjoint. Suurballe [10] presented polynomial time algorithms for computation of a pair of disjoint paths such that the sum of the path lengths is minimum. The idea of using intermediate nodes to provide a level of indirection in creating an alternate path was proposed in [1, 5, 12, 2]. These intermediate nodes have been referred with various nomenclature: as *overlay nodes* [1, 5], *rendezvous points* [12], *depots* [2], *hubs* [4] and *transit hubs* in this paper.

Our first contribution is to establish the NP-completeness of the K-transit hub routing problem. Secondly, we present an exact algorithm for solving the K-Transit hub routing problem. Our third contribution is a heuristic based solution that is effective for large networks. We evaluate the efficacy of our heuristic through extensive experimentation on *PlanetLab's Abilene* network and various randomly generated topologies.

2 Disjoint Path Routing Using Transit Hubs

The objective of the K-Transit hub routing problem is to find out if it is possible to construct a path from s to d by concatenating at most $K + 1$ paths (from the set of $n * (n - 1)$ paths) so that (i) each of these paths is edge-disjoint with the original s to d path and (ii) the paths are mutually edge-disjoint. In other words, can we find a set of paths $\{P_{s,v_1}, P_{v_1,v_2}, P_{v_2,v_3}, P_{v_3,v_4}, \ldots, P_{v_{k-1},v_k}, P_{v_k,d}\}$, $1 \leq k \leq K + 1$ such that the concatenation of these paths will produce a path from s to d and condition (i), (ii) above are satisfied.

The idea is illustrated with the help of an example overlay network (Fig. 1) obtained from the PlanetLab. The overlay network has five nodes 1 through 5. The nodes a through j represent routers through which the overlay nodes establish paths between each other. In this example, the primary path for data transfer from overlay node 1 to node 4 is through the link 1-4 (path P_3). If the following question is asked: *"Is it possible to construct an alternate path from node 1 to 4, disjoint from the default path, by concatenating at most two mutually*

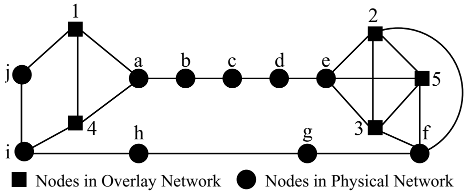


Fig. 1. Overlay Network from the PlanetLab (See legend)

Legend: Nodes in Fig 1	**Paths connecting overlay nodes**
1: PlanetLab 1, University of Arizona	P1: 1 - a - b - c - d - e - 2;
2: PlanetLab 2, Carnegie Mellon University	P2: 1 - a - b - c - d - e - 3;
3: PlanetLab 2, Duke University	P3: 1 - 4; P4: 1 - a - b - c - d - e - 5;
4: PlanetLab 2, University of Washington	P5: 2 - f - g - h - i - j - 1; P6: 2 - 3;
5: PlanetLab 2, Princeton University	P7: 2 - f - g - h - i - 4; P8: 2 - 5;
a: kscyng-dnvrng.abilene.ucaid.edu	P9: 3 - f - g - h - i - j - 1;
b: iplsng-kscyng.abilene.ucaid.edu	P10: 3 - 2;
c: chinng-iplsng.abilene.ucaid.edu	P11: 3 - f - g - h - i - 4; P12: 3 - 5;
d: nycmng-chinng.abilene.ucaid.edu	P13: 4 - 1;
e: washng-nycmng.abilene.ucaid.edu	P14: 4 - a - b - c - d - e - 2;
f: nycmng-washng.abilene.ucaid.edu	P15: 4 - a - b - c - d - e - 3;
g: chinng-nycmng.abilene.ucaid.edu	P16: 4 - a - b - c - d - e - 5;
h: iplsng-chinng.abilene.ucaid.edu	P17: 5 - f - g - h - i - j - 1;
i: kscyng-iplsng.abilene.ucaid.edu	P18: 5 - 2; P19: 5 - 3;
j: dnvrng-kscyng.abilene.ucaid.edu	P20: 5 - f - g - h - i - 4;

disjoint paths?", the answer to the question is *"yes"* because such a path can be constructed by concatenating paths P_4 and P_{20}.

3 Problem Formulation and Complexity Analysis

As indicated earlier, the input to the K-Transit hub routing problem is (i) an undirected network graph $G = (V, E)$, (ii) a set of $n * (n - 1)$ paths ($|V| = n$) between every source-destination node pair (the path from node i to j may notbe same as the path from j to i) and (iii) specified source and destination nodes s and d respectively. The objective of the K-Transit hub routing problem is to find out if it is possible to construct a path from s to d by concatenating at most $K + 1$ paths (from the set of $n * (n - 1)$ paths) so that (i) each of these paths is edge-disjoint with the original s to d path and (ii) the paths are mutually edge-disjoint.

In order to find an answer to this question, we first remove all the edges used by the path from s to d from the graph $G = (V, E)$. Let $\mathcal{P}$ be the set of all $n * (n - 1)$ paths given as the input. After removal of the edges belonging to the s to d path, many of the paths in $\mathcal{P}$ may become disconnected. We refer to such paths as *"unavailable"* (P_{unav}). The other subset of paths in $\mathcal{P}$ are the *"available"* paths (P_{av}).

Note that here we make no attempt to consider future connection requests and the bandwidth required to satisfy them. It could very well happen that the alternate path computed by our algorithms may not be able to satisfy a connection request due to other network traffic at that time. In our approach, we merely try to ascertain whether such an alternate path exists in the network. The rationale behind this decision is that the K-Transit hub problem by itself without considering any of these parameters is NP-complete. In order to keep the problem tenable, we focus on just the computation of an alternate path by concatenating at most $K + 1$ paths. In this regard, we do not consider the capacity of the links in our model.

3.1 Definitions and Notations

Definition 1. Intersection set of paths: *The intersection set of two paths P_i and P_j is the set of edges common between the paths and is denoted by $P_i \cap P_j$.*

Definition 2. Compatible Paths*: Two paths P_i and P_j are said to be compatible if their intersection set is empty.*

Definition 3. Concatenation of Paths*: If P_i is a path from s_i to d_i and P_j is a path from s_j to d_j, they can be concatenated if $d_i = s_j$ and the result of the concatenation operation is a path from s_i to d_j.*

Definition 4. K-Transit Hub Routing Problem
Instance: Given an undirected graph $G = (V, E)$, a set of triples $(s_i, d_i, P_i), 1 \leq i \leq r$, where s_i is a source node, d_i is destination node and P_i is a path from s_i to d_i and r is the number of such triples, specified source, destination nodes s and d respectively and an integer K.

Question: Suppose $\mathcal{P}_{av} = \{P_1, \ldots, P_r\}$. Is there a subset $\mathcal{P}'_{av} \subseteq \mathcal{P}_{av}$ such that:

(i) $\mid \mathcal{P}'_{av} \mid \leq K + 1$
(ii) The paths in $\mathcal{P}'_{av}$ are mutually compatible, i.e., if $P_i, P_j \in \mathcal{P}'_{av}$, then $P_i \cap P_j = \emptyset, \forall i \neq j$ and
(iii) A path from s to d can be constructed by concatenating the paths in $\mathcal{P}'_{av}$.

3.2 Complexity Analysis

Theorem. The K-Transit Hub Routing Problem is NP-Complete.

Proof. It is not difficult to verify that the K-Transit hub routing problem is in NP. We show that the K-Transit Hub Routing Problem is NP-complete by a polynomial transformation from the 3SAT problem. From a given instance of the 3SAT problem, specified by a set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a set of clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$, we construct an instance of the K-Transit Hub Routing Problem in the following way. First, we classify each edge in the graph $G = (V, E)$ as a path-edge or a non-path-edge.

Definition 5. *An edge $(u, v) \in E$ is called a path-edge, if $\exists P_i = (u, v)$ where $P_i \in \mathcal{P}_{av}$. Otherwise, the edge is known as a non-path-edge.*

It may be noted that a path between a source-destination node pair may comprise of both of these types of edges. The instance $(G, \mathcal{P}_{av}, s, d, K)$ of the K-Transit hub routing problem can be generated from the instance of the 3-SAT problem in three steps: (i) We construct a subgraph G_1 of G. (ii) Paths in $\mathcal{P}_{av}$ consisting of more than one edge are specified. (iii) We augment G_1 with additional nodes and edges to construct G. It may be noted that all paths in $\mathcal{P}_{av}$ consisting of exactly one edge are specified in (i) and (iii), and all paths with more than one edge are specified in (ii).

Step 1. $\forall x_i \in \mathcal{X}$ and $\forall C_j \in \mathcal{C}$, construct a 4-node subgraph with node set $\{u_{i,j}, u'_{i,j}, v_{i,j}, v'_{i,j}\}$ and edge set $\{(u_{i,j}, u'_{i,j}), (v_{i,j}, v'_{i,j})\}$, where both edges are path-edges. For each fixed x_i, $\forall j = 1, \ldots, m-1$, we connect the subgraph for x_i, C_j and the one for x_i, C_{j+1} with two non-path-edges $(u'_{i,j}, u_{i,j+1})$ and $(v'_{i,j}, v_{i,j+1})$. Then for each x_i, we add six more vertices: $a_i, b_i, c_i, a'_i, b'_i$ and c'_i. We connect a_i, b_i, c_i with path-edges (a_i, b_i) and (a_i, c_i) and connect a'_i, b'_i, c'_i with path-edges (a'_i, b'_i) and (a'_i, c'_i). For each x_i, we add four more non-path-edges: $(b_i, u_{i,1}), (c_i, v_{i,1}), (u'_{i,m}, b'_i)$ and $(v'_{i,m}, c'_i)$. In addition, $\forall i = 1, \ldots, n-1$, we add a path-edge (a'_i, a_{i+1}) to connect the subgraphs corresponding to x_i and x_{i+1}. If the instance of the 3SAT problem is given by $\phi = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$, then the graph G_1 corresponding to ϕ is shown in Fig. 2.

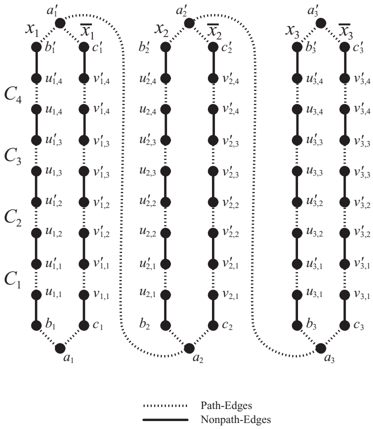


Fig. 2. Subgraph G_1 of G

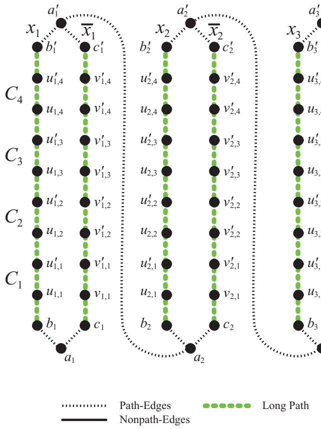


Fig. 3. Long paths consisting of more than one edge

Step 2. In this step, we specify all the paths in $\mathcal{P}_{av}$ consisting of more than one edge. $\forall i = 1, \ldots, n$, a path between b_i and b'_i: $P_{b_i} = b_i - u_{i,1} - u'_{i,1} \ldots - u'_{i,m} - b'_i$ and a path between c_i and c'_i: $P_{c_i} = c_i - v_{i,1} - v'_{i,1} \ldots - v'_{i,m} - c'_i$ are added into $\mathcal{P}_{av}$. For the example used in Step 1, the paths specified in this step are highlighted in Fig. 3.

Step 3. This step has two parts. First, a set of nodes $\{s, w_0, w_1, \ldots, w_m, d\}$ is added to the graph G_1 (Recall that m is the number of clauses in the 3SAT instance). Second, a set of path-edges is added as follows: (i) connect s and w_0 by a path-edge (s, w_0), connect a'_n and d by a path-edge (a'_n, d) (ii) $\forall i = 1, \ldots, n, \forall j = 1, \ldots, m$, if $x_i \in C_j$, then add $(w_{j-1}, u_{i,j})$, $(u'_{i,j}, w_j)$ and if $\bar{x}_i \in C_j$, then add $(w_{j-1}, v_{i,j})$, $(v'_{i,j}, w_j)$ (iii) connect w_m to a_1 by a path-edge (w_m, a_1). This completes the construction procedure of G with all of paths in $\mathcal{P}_{av}$. The resulting graph G is shown in Fig. 4.

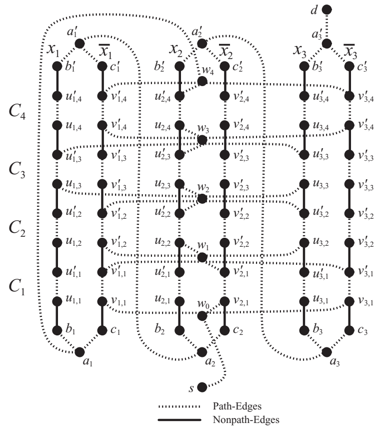


Fig. 4. Graph G

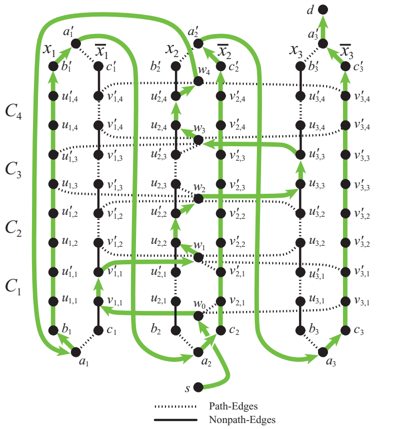


Fig. 5. The transit hub path corresponding to the truth assignment of ϕ

Set s, d to be the source node and destination node respectively in graph $G = (V, E)$.Set K=$|E|$. Construction of the instance of the K-Transit hub problem is now complete.

Claim: There exists a truth assignment satisfying the instance of the 3SAT problem, if and only if a path from s to d can be constructed in the generated instance of the K-Transit hub routing problem by concatenating at most $K + 1$ mutually compatible paths.

Proof of the claim: Suppose that there is a truth assignment satisfying the instance of the 3SAT problem. We can construct a path from s to d by concatenating a subset of paths in the following way: (i)Go from s to w_0 following the path-edge between them. (ii)Each $C_j, j = 1, \ldots, m$, has at least one literal, z that has been assigned "true" by the truth assignment. This implies that we can go from w_{j-1} to w_j using the corresponding path-edges (i.e., $w_{j-1} - u_{i,j} - u'_{i,j} - w_j$ or $w_{j-1} - v_{i,j} - v'_{i,j} - w_j$). (iii)Go from w_m to a_1 using the path-edge between them. (iv)If $x_1 =$ "true", then no edge on the path from c_1 to c'_1 has been used so far; otherwise, if $x_1 =$ "false", then no edge on the path from b_1 to b'_1 has been used yet. Hence, we can go from a_1 to a'_1 using one of the following two sequences of paths: if the path from b_1 to b'_1 is unused, then take (a_1, b_1), path from b_1 to b'_1, and then (b'_1, a'_1); if the path from c_1 to c'_1 is unused, then take (a_1, c_1), path from c_1 to c'_1, and then (c'_1, a'_1). (v)$\forall i, 1 \leq i \leq n-1$, go from a'_i to a_{i+1}, using the path-edge between them. (vi)$\forall i, 2 \leq i \leq n$ go from a_i to a'_i following the same process as in step (iv). (vii)Go from a'_n to d following the path-edge between them. Thus, we find a $s - d$ path, which is a concatenation of a sequence of mutually compatible paths.

To prove the converse, suppose that we can go from s to d by concatenating a sequence of mutually compatible paths. It is not hard see that we must first go from s to w_m by following the path-edges incident to w_j's. Then, from w_m, we have to go through each subgraph corresponding to x_i's, from a_i to a'_i, by using the long paths from b_i to b'_i, or the ones from c_i to c'_i. $\forall i = 1, \ldots, n$, if the path from b_i to b'_i is used, then assign x_i to be "false" and if the path from c_i to c'_i is used, then assign x_i to be "true". It is not hard to check this assignment satisfies the corresponding 3SAT problem. This completes the proof of the theorem.□

In the sample 3SAT instance ϕ considered in Steps 1, 2, and 3, the truth assignment $f(x_1) = FALSE$, $f(x_2) = TRUE$, $f(x_3) = TRUE$ satisfies ϕ. The corresponding $s - d$ path is shown in Fig. 5

4 Exact Solution for the K-Transit Hub Routing Problem

In this section, we provide an exact algorithm for the solution of the K-Transit hub routing problem. As a first step in that direction, we first construct a *Path Intersection Graph (PIG)*.

Definition 6. *A* Path Intersection Graph *is the intersection graph of paths in the set $\mathcal{P}_{av}$. This is a graph $G_{pig} = (V_{pig}, E_{pig})$, where each node represents a path in the set $\mathcal{P}_{av}$ and two nodes have an edge between them, if the corresponding paths have any common edge.*

Definition 7. *An independent set (or a stable set) in a graph $G = (V, E)$ is a subset $V' \subseteq V$, such that no two nodes in V' are adjacent to each other in the graph $G = (V, E)$.*

Definition 8. *An independent set in a graph $G = (V, E)$ is called a* maximal independent set *if it is not a proper subset of any other independent set in the graph.*

As a second step towards construction of the alternate s to d path, we compute all the maximal independent sets of the path intersection graph. The maximal independent sets of the path intersection graph will correspond to the sets of maximal compatible paths in $\mathcal{P}_{av}$. Let $\{MIS_1, MIS_2, \ldots\}$ represent the set of maximal independent sets of the path intersection graph.

As a third step in the process to construct an alternate s to d path, we construct a *Path Construction Graph* corresponding to each maximal independent set $MIS_1, MIS_2, \ldots, MIS_t$ computed in the previous step.

Definition 9. *Each node in a* Path Construction Graph *corresponding to a* $MIS_i, 1 \leq i \leq t$, $G_{pcg}(i) = (V_{pcg}(i), E_{pcg}(i))$, *corresponds to a path in* MIS_i *and two nodes have an edge between them if the corresponding paths have a common terminating point, i.e., if the terminating points of a path are* v_i, v_j *and the terminating points of another path are* v_k, v_j, *then the nodes corresponding to these two paths will have an edge between them in the graph* $G_{pcg}(i)$.

Let $V_{pcg}(i, s) = \{v_{s,1}, v_{s,2}, \ldots, v_{s,p}\}$ denote the set of nodes that correspond to paths whose one terminating point is the designated source node s. Similarly, let $V_{pcg}(i, d) = \{v_{d,1}, v_{d,2}, \ldots, v_{d,q}\}$ denote the set of nodes that correspond to paths whose one terminating point is the designated destination node d. Now in the graph $G_{pcg}(i)$, we compute the shortest path between the nodes $v_{s,j}, 1 \leq j \leq p$ and $v_{d,k}, 1 \leq k \leq q$. If any of these paths have length at most $K + 1$, then it is *possible* to construct an alternate path from s to d, disjoint from the original path $P_{s,d}$ in the graph $G = (V, E)$, by concatenating compatible paths in the set $\mathcal{P}_{av}$. This process of building a path construction graph $G_{pcg}(i)$ from MIS_i followed by the computation of shortest path needs to be repeated $\forall i, 1 \leq i \leq t$, where t is the number of maximal independent sets. If a shortest path of length at most $K + 1$ cannot be found in any one of these graphs $G_{pcg}(i), 1 \leq i \leq t$, then it is *impossible* to construct an alternate path from s to d, disjoint from the original path $P_{s,d}$ in the graph $G = (V, E)$ by concatenating compatible paths in the set $\mathcal{P}_{av}$.

4.1 Algorithm Analysis

The algorithm first computes the Path Intersection Graph of the set of available paths $\mathcal{P}_{av}$ and then computes all maximal independent sets of this graph. The maximal independent sets give the set of compatible paths that can be concatenated for constructing the path from the source s to destination d. In step 5 of the algorithm the Path Construction Graph is constructed and in step 7, the shortest path between a $v_{i,s}$ and $v_{i,d}$ is computed. Since the process is repeated for all maximal independent sets that contains a $v_{i,s}$ and $v_{i,d}$ and for all $v_{i,s}$ and $v_{i,d}$, if a path between s to d can be obtained by concatenating at most $K+1$ compatible paths in the set $\mathcal{P}_{av}$, this process will find it. This ensures the correctness of the algorithm.

For generating all maximal independent sets of a graph, algorithms such as the ones presented in [11] and [6] can be used. Both the algorithms produce the maximal independent sets one after another in such a way that the delay

Algorithm 1. K-Transit Hub Routing Exact Solution$(G, \mathcal{P}_{av}, s, d, K)$

```
step_1 Compute Path Intersection Graph, $G_{pig} = (V_{pig}, E_{pig})$ for the paths in $\mathcal{P}_{av}$.
step_2 Compute all Maximal Independent Sets of $G_{pig}$, $\mathcal{MIS} = \{MIS_1, MIS_2, \ldots, MIS_t\}$.
step_3 Compute a subset $\mathcal{MIS}' \subseteq \mathcal{MIS}$, such that all elements of $\mathcal{MIS}'$, contain
       at least one path whose terminating point is $s$ and another path whose terminating point is $d$.
step_4 Repeat steps 5-7 for each elements $MIS_i$ of $\mathcal{MIS}'$,
step_5 Compute the Path Construction Graph $G_{pcg}(i)$ corresponding to $MIS_i$
step_6 Let $V_{i,s}$ be the set of nodes in $G_{pcg}(i)$ that corresponds to those paths whose
       one terminating point is $s$ and $V_{i,d}$ be the set of nodes in $G_{pcg}(i)$ that corresponds
       to those paths whose one terminating point is $s$. Repeat step 5 for
       each element $v_{i,s} \in V_{i,s}$ and for each element $v_{i,d} \in V_{i,d}$
step_7 Compute the shortest path from $v_{i,s}$ to $v_{i,d}$. If the shortest path length is at
       most $K + 1$, then an alternate path from $s$ to $d$ using compatible paths from
       the set $\mathcal{P}_{av}$ exists. EXIT from the loop.
step_8 If no path of length at most $K + 1$ can be found in any of the combinations of
       $v_{i,s}$ and $v_{i,d}$, then an alternate path from $s$ to $d$ using compatible paths from
       the set $\mathcal{P}_{av}$ does not exist.
step_9 EXIT
```

between generation of two consecutive maximal independent sets is bounded by a polynomial function of the input size. The computation complexity of the algorithm in [11] is $O(n * m * \alpha)$ and the algorithm in [6] is $O(n^3 * \alpha)$ where n, m and α represents the number of nodes, edges and the maximal independent sets of the graph respectively. We use the algorithm in [6] for generating all maximal independent sets in step 2 of the K-Transit hub routing algorithm.

Let α, β represent the number maximal independent sets of the path intersection graph and the paths (i.e. $|\mathcal{P}_{av}|$) respectively. The worst case computational complexity of step 1 of the algorithm is $O(\beta^2)$, step 2 is $O(\beta^3 * \alpha)$ and step 3 is $O(\beta^2 * \alpha)$. Thus, the overall complexity of the algorithm is $O(\alpha * \beta^4)$.

5 Heuristic Solution for the *K*-Transit Hub Routing Problem

The main overhead involved in the exact algorithm is in the computation of all the maximal independent sets of the path intersection graph. In this section, we present a heuristic solution using randomization technique for the K-Transit Hub Routing problem that produces a solution with high probability. The complexity of the solution is bounded by a polynomial function of the number of nodes in the overlay network.

5.1 Complexity Analysis

As in the exact algorithm, the heuristic solution starts by determining the Path Intersection Graph of the available paths $\mathcal{P}_{av}$. However, instead of finding all

Algorithm 2. Heuristic for K-Transit Hub Routing Problem $(G, \mathcal{P}_{av}, s, d, K)$

```
step_1 Compute Path Intersection Graph $G_{pig} = (V_{pig}, E_{pig})$ for paths in $P_{av}$.
step_2 Compute set of nodes, $V_s \in V_{pig}$ that correspond to the paths whose one terminating point is $s$ and $V_d \in V_{pig}$ as nodes that correspond to the paths whose one termination point is $d$.
step_3 Repeat steps 4 through 7 for every node-pair $(v_s, v_d) \in V_s \times V_d$.
step_4 Construct a maximal independent set with two nodes $v_s$ and $v_d$, $\mathcal{MIS} = \{v_s, v_d\}$
step_5 Let $\mathcal{NNS}(S)$, the "Non-Neighborhood Set" of $S$ be defined as $\mathcal{NNS}(S) = V_{pig} \backslash (\mathcal{N}(S) \bigcup S)$, where $\mathcal{N}(S)$ represents the neighborhood set of $S$. Select with equal probability a node $v \in \mathcal{NNS}(S)$. Augment the maximal independent set, $\mathcal{MIS} = \mathcal{MIS} \bigcup \{v\}$.
step_6 If $\mathcal{MIS}$ is not a maximal independent set, go back to step_5. Otherwise, form the Path Construction Graph $G_{pcg}$. Compute $V_{i,s}, V_{i,d} \in V(G_{pcg})$ as the set of nodes corresponding to paths having one terminating point in $s, d$ respectively.
step_7 Compute the shortest path between every pair $v_{i,s} \in V_{i,s}$ and $v_{i,d} \in V_{i,d}$. If there exists a shortest path of length at most $K + 1$ between any $v_{i,s} \in V_{i,s}$ and $v_{i,d} \in V_{i,d}$, then an alternate path from $s$ to $d$ using compatible paths from the set $\mathcal{P}$ exists. EXIT from the loop.
step_8 If none of the combinations of $v_s$ and $v_d$ report a path of length at most $K+1$, then there is no alternate path from $s$ to $d$ using compatible paths from the set $\mathcal{P}_{av}$.
step_9 EXIT
```

maximal independent sets involving the two nodes(paths) v_s and v_d that terminate in s and d respectively, the algorithm randomly generates a maximal independent set for each pair-wise combination of v_s and v_d. The random generation procedure first includes the two nodes v_s and v_d into a working set $\mathcal{MIS}$ of independent nodes. It then randomly selects a node from all the remaining non-neighboring nodes of $\mathcal{MIS}$ in the Path Intersection Graph and includes it into $\mathcal{MIS}$. This process is continued until $\mathcal{MIS}$ is maximally independent.

Let β is the number of nodes in the Path Intersection Graph. Step 1 has worst case computational complexity of $O(\beta^2)$. Steps 5 and 6 perform $O(\beta^2)$ operations in the worst case to compute a Maximal Independent Set. Step 7 of the algorithm performs $O(\beta^2)$ operations to check if there exist compatible paths in the Path Construction Graph between the source node and the destination node. Thus, the overall complexity of the algorithm is $O(\beta^4)$.

5.2 Performance of the Heuristic Solution

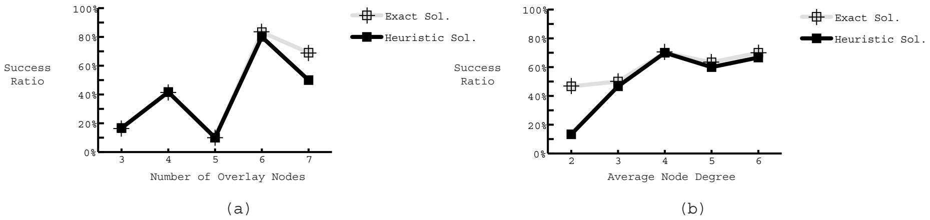
To evaluate the performance of our proposed exact and heuristic solutions, we conducted experiments for the K-transit hub routing problem on randomly generated topologies and the Abilene network.

The problem instances were generated in 3 steps:

Step 1. Georgia-Tech Internet Topology Model topology generator was used to generate the random physical layer topologies having 30 nodes and average node degree varying between 2 and 6.
Step 2. A subset of these nodes were randomly chosen with uniform distribution as the set of overlay nodes.
Step 3. Shortest Paths using Dijkstra's algorithm between every pair of the overlay nodes were computed. These act as the primary paths in our experiments.

One of the metrics used for the evaluation of the performance of the heuristic is the *success ratio*. *Success ratio* is the ratio of number of source-destination pairs for which a path was found by the algorithm to total number of source-destination pairs.

Three sets of experiments were conducted to study the performance of the heuristic solutions. In the first set (6(a)), the number of overlay nodes were varied from 3 to 7 and success ratio of both the exact and the heuristic solutions were measured for all source-destination pairs. The value of K was chosen to be greater than the number of overlay nodes. In the second set of experiments 6(b), different physical topologies consisting of 30 nodes were chosen with varying average node degrees. In each case, 6 nodes were chosen to be overlay nodes and the success ratio of the exact and heuristic algorithms were measured for all the 30 source-destination pairs. The aim of this experiment was to study the impact of the average node degree on the performance of the algorithm. The third set of experiments (7) were conducted with two data sets. For various values of K, the success ratio of both the algorithms were recorded. The physical topology had 30 nodes with an average node-degree of 4 and the overlay structure had 7 nodes.



(a)

(b)

Fig. 6. Performance of the Heuristic Solution, (a) Success Ratio vs. Number of overlay nodes; (b) Success ratio vs. Average node degree

In most of the cases, the success ratio of the heuristic was close to the exact algorithm. Increasing average node-degree in the physical topology (6(b)) has a positive effect on finding alternate paths in the overlay. The success ratio for both the heuristic and exact algorithms increase with increased average node-degree. The results (7(a), (b)) indicate that the performance of the heuristic solution is not significantly dependent on the value of K, the number of paths that are

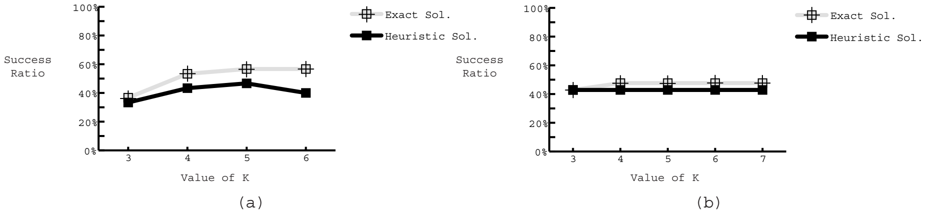


Fig. 7. Performance of the Heuristic Solution, (a) Success ratio vs. Value of K for instance 1; (b) Success ratio vs. Value of K for instance 2

allowed to be concatenated to construct the source to destination path. In all these experiments, the execution times of the heuristic and exact solution were noted. In many instances, the execution time of the exact solution was almost 1000 times more than that of the heuristic. We thus conclude that our heuristic technique almost always produces a very high quality solution in a fraction of time needed to find the exact solution.

6 Conclusion

In this paper, we consider the problem of computing an alternate path that is disjoint to the default IP path. Such an alternate path can be computed by exploiting transit hubs placed at opportunistic locations on the Internet. We show that the problem of finding such a path with constraint on the number of transit hubs is NP-complete. We provide an exact and approximate solution for the problem. Our experimentations demonstrate that our heuristic produces near optimal solution for most of the instances in a fraction of time needed to find the optimal solution.

References

1. D. Anderson, H. Balakrishnan, M. Kaashoek and R. Morris, “Resilient Overlay Networks,” *In Proc. 18th ACM SOSP,* Canada, October 2001.
2. M. Beck, J. Dongarra, J. Plank and R. Wolski, Logistical Network Project, *http://loci.cs.utk.edu/scidac*.
3. I. Cidon, R. Rom and Y. Shavitt, Analysis of Multi-path Routing, *IEEE/ACM Trans. on Networking*, vol. 7, no. 6, pp. 885-896, 1999.
4. R. Cohen and G. Nakibli, On the computational complexity and effectiveness of “N-hub shortest path routing”, *Proc. of IEEE Infocom*, 2004.
5. N. Feamster, D. Anderson, H. Balakrishnan and M. Kaashoek, “Measuring the Effects of Internet Path Faults on Reactive Routing,” *In Proc. of ACM SIGMETRICS,* San Diego, CA, June 2003.
6. D. S. Johnson, M. Yannakakis and C. H. Papadimitriou, On Generating All Maximal Independent Sets, *Information Processing Letters*, vol 27, pp. 119-123, 1988.

7. T. Kosar, G. Kola and M. Livny, A Framework for Self-optimizing, Fault-tolerant, High Performance Bulk Data Transfers in a Heterogenous Grid Environments, *Proc. ISPDC* 2003.
8. S. Lee and M. Gerla, Split Multipath Routing with Maximally Disjoint Paths in Ad-hoc Networks, *Proc. of IEEE ICC* 2001.
9. C. Perkins, "IP encapsulation within IP," *IETF RFC 2003,* October 1996.
10. S. Suurballe and R. Tarjan, A Quick Method for Finding Shortest Pair of Disjoint Paths, *Networks*, vol. 14, pp. 325–336, 1984.
11. S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A New Algorithm for Generating All Maximal Independent Sets, *SIAM Journal of Computing*, vol. 6, pp. 505-517, 1977.
12. S. Z. S. S. I. Stoica, D. Adkins and S. Surana, "Internet indirection infrastructure," *in Proceedings of ACM SIGCOMM 2002,* August 2002.

Comparison of Border-to-Border Budget Based Network Admission Control and Capacity Overprovisioning[*]

Ruediger Martin[1], Michael Menth[1], and Joachim Charzinski[2]

[1] Department of Distributed Systems, Institute of Computer Science,
University of Würzburg, Am Hubland, D-97074 Würzburg, Germany
{martin, menth}@informatik.uni-wuerzburg.de
[2] Siemens AG, Munich, Germany
joachim.charzinski@siemens.com

Abstract. There are two basic approaches to achieve Quality of Service (QoS) for communication networks: admission control (AC) and capacity overprovisioning (CO). CO is simple and cheaper to implement than AC but AC requires less capacity to fulfill QoS criteria since overload traffic can be blocked. There is an almost religious war between scientists working on both concepts. In this paper we try to contribute insights for this discussion by quantifying the capacity savings potential of AC under various networking conditions.

Keywords: QoS, admission control, capacity overprovisioning.

1 Introduction

The traditional Internet offers a best effort service and almost global reachability at low cost. Bulk transfers require high throughput, value added services like telephony or video conference depend on short packet delays and predictable throughput, and precision applications like tele-medicine or tele-robotics cannot even afford packet loss. Therefore, Quality of Service (QoS) in terms of short packet delay and low packet loss is required for next generation networks (NGNs) to support these services.

QoS can be achieved by limiting the traffic volume in the network by admission control (AC) and thereby preventing overload situations. As an alternative, sufficient capacity can be provisioned such that no congestion occurs. This is called capacity overprovisioning (CO). On the one side, many investigations compare blocking probabilities of different AC schemes for which several signalling protocols exist. On the other side, practical experience shows that CO is already applied since the utilization of core networks today is very low [1].

In this paper we quantify the capacity requirements for networks that rely on AC and CO for QoS provisioning, which is crucial for an economic assessment of both ap-

[*] This work was funded by the Bundesministerium für Bildung und Forschung of the Federal Republic of Germany (Förderkennzeichen 01AK045) and Siemens AG, Munich. The authors alone are responsible for the content of the paper.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1056–1068, 2005.

proaches. We focus on the simplest method for network AC (NAC), namely on border-to-border (b2b) budget (BBB) based NAC. We consider networks with only high priority real-time traffic which is not elastic and which has a connection structure. This is a typical scenario in, e.g., core networks for cellular systems. We consider first an unrealistically simple static traffic model on a single link. We add overload situations and extend the study to entire networks such that our final results reflect realistic scenarios.

The paper is structured as follows. In Sec. 2, we give a short introduction to AC and CO, discussing related work and our assumptions. Section 3 develops a traffic model and suggests capacity dimensioning methods for AC and CO both for a single link and for entire networks. In Sec. 4, we present the capacity requirements for AC and CO under various networking conditions. Sec. 5 discusses the results and our conclusions.

2 Overview on Admission Control and Capacity Overprovisioning

We give an overview on various aspects of AC by focusing first on the packet level and then on the flow level. Then we consider related work regarding CO and find a suitable level on which we can compare AC and CO.

2.1 Admission Control

QoS can be defined by a loss and a delay parameter. The packet loss probability should be smaller than, e.g., 10^{-6} and the 99.99%-percentile of the waiting time should not exceed a given delay budget DB, i.e., the probability for a packet to wait longer than DB must be smaller than 0.01%. This is achieved by limiting the traffic per transmission resource to avoid overload, i.e., flows request admission for their transportation over certain resources which can be granted or denied. We identify AC methods for a single resource that we call link AC (LAC) and methods that coordinate several resources which we call network AC (NAC). An extensive overview on AC can be found in [2].

Link Admission Control. Link AC (LAC) methods concentrate primarily on the packet level and on a single resource. Thus, traffic descriptors characterize the packet streams by token bucket or dual token bucket parameters to capture the variability of the traffic on two different time scales. They inform the AC entity and the policer about a maximum peak rate and inter-packet distance. This information is used to calculate together with other assumptions the packet loss probability and the expected delay distribution on the link. A generalization and simplification of that approach is the concept of effective bandwidth [3]. It depends on such traffic descriptors and other parameters like the link capacity and assigns a so-called effective bandwidth to any flow request. If the effective bandwidth sum of admitted flows plus the effective bandwidth of a new request exceeds a certain capacity budget, e.g. the link capacity, then the flow is rejected; otherwise it is accepted. Thus, we get the flow blocking probability p_b as additional measure for GoS.

Network Admission Control. Network AC (NAC) methods concentrate on AC for several resources, e.g., for a path consisting of several single links within a network.

The link budget (LB) based NAC performs the above described AC successively on each link of the flow's path and it is successful if all AC decisions are positive. This is the most intuitive NAC approach and it has been implemented by many signalling protocols, e.g. by RSVP. The drawback of this method is that information about flows must be kept not only by the ingress router but also by all routers along the path. This increases the administration overhead and it complicates a resilient architecture. The border-to-border (b2b) budget (BBB) based NAC defines capacity budgets for each b2b relationship (v, w) within the network and assigns them a capacity portion. A new flow originating at ingress border router v and destined for egress border router w asks for admission only at its ingress router v. This ingress router performs AC based on $BBB(v, w)$ like on a single resource. This AC type has been enhanced by resilience mechanisms and it has been successfully implemented within the KING project [4]. Another example for BBB NAC is AC based on label switched paths (LSPs) with fixed capacity. In this work, we compare the required capacity for BBB NAC and CO.

2.2 Capacity Overprovisioning

Capacity overprovisioning (CO) purely relies on provisioning enough bandwidth to meet a desired QoS. The QoS definition from above in terms of packet loss probability and delay budget still holds. As CO does not limit the traffic to avoid overload, all flows are admitted. The link capacities are chosen such that they are very rarely exceeded by the predicted traffic. Like AC, CO can also be combined with different traffic classes by implementing priority scheduling mechanisms. Low priority traffic can use the bandwidth provisioned for high priority traffic under non-overlad situations without additional mechanisms.

Related Work. Bandwidth provisioning procedures differ fundamentally from access to core networks due to the degree of aggregation. Empirical evidence can be found in [5, 6] that core network traffic on the packet level, i.e. the average traffic arrival rate, is modeled well by the Gaussian distribution due to the high level of aggregation. This is clearly not the case in the access due to the limited number of users where the aggregation level is inherently low.

A comparison of AC and CO in access network dimensioning is the topic of [7]. The authors find a clear benefit of AC. Depending on network parameters like blocking probability, packet loss probability and user activity, the number of subscribers for a given access network capacity is substantially higher when AC is used. However, we focus on core networks.

In [5] the network is dimensioned to support latency sensitive traffic. Accordingly, the QoS measure the network is dimensioned for is the probability that the queue length Q of a router exceeds a certain value x: $\mathrm{P}\{Q > x\}$. End-to-end delay requirements of 3ms require only 15% extra bandwidth above the average data rate of the traffic in the highly aggregated Sprint network. Another approach [8] focuses on the probability that the amount of traffic $A(T)$ generated on a link within a specified time interval T exceeds the capacity C of the link: $\mathrm{P}\{A(T) \geq C \cdot T\}$. The authors argue that applications can cope with lack of bandwidth within an application-dependent small interval T if this occurs sufficiently rarely. They develop an interpolation formula that predicts the bandwidth requirement on a relatively short time scale in the order of 1 second by relying on coarse traffic measurements.

Another closely related problem is forecasting of Internet traffic. A recent approach for long-term forecasting can be found in [9]. The authors of [10] combine both tasks to yield an adaptive bandwidth provisioning algorithm. Based on measurements, the required capacity is predicted and adjusted on relatively small time scales between $4s$ and $2min$. The Maximum Variance Asymptotic (MVA) approach for the tail probability of a buffer fed by an Gaussian input process is used to make the QoS requirement $\mathrm{P}\{delay > D\} < \epsilon$ explicit.

Our work is different from the literature presented here. Our focus is not the development of an AC or CO scheme to adjust the network capacity to the needs of a specific application or specific scenario. We develop a model to qualitatively compare the required capacity for AC versus the required capacity for CO.

Our View on CO for Comparison with AC. Most CO studies use both a flow and a packet level model. The first models the number of flows in the network whereas the second causes the required extra bandwidth above the mean data rate of the traffic.

Both AC and CO can be combined with a packet level model to asses the relation of effective bandwidth to average and peak data rates, and an inadequate packet level model will lead to QoS degradations in both systems.

However, we are primarily interested in comparing AC to CO and not in specific statistical multiplexing schemes. Therefore, we eliminate the packet level by working on effective bandwidths for both systems. This is a prerequisite for a fair comparison.

If the requested rate of all flows on a link exceeds the link bandwidth, all flows are affected by QoS degradation. This view leads to the definition of a new QoS measure for CO: the QoS violation probability p_v which is the time fraction with violated QoS. As all flows are concerned, it should be low and we use an objective value of $p_v = 10^{-6}$ for bandwidth dimensioning in our study.

3 Capacity Dimensioning

Now we describe the capacity dimensioning methods used for AC and CO both for a single link and an entire network.

3.1 Traffic Model

Real-time flows are mostly triggered by human beings. Thus, their inter-arrival time is exponentially distributed [11]. The Poisson model for flow arrivals is also advocated by [12] and current evidence of Poisson inter-arrivals for VoIP call arrivals is given in [13]. Therefore, a flow level model that is characterized by exponentially distributed inter-arrival time and an independently and identically distributed call holding time is appropriate in an evolving multimedia world.

Multi-rate Traffic As the request profile is multi-rate in a multi-service network like the Internet, we use a simplified multi-rate model (cf. [2]). We have $n_r = 3$ different request types r_i, $0 \leq i < n_r$ with request sizes $c(r_i) \in \{64, 256, 2048\}$ kbit/s. The mean

Table 1. Request type statistics

request type r_i	$c(r_i)$	$P\{C_t = c(r_i)\}$
r_o	64 kbit/s	$\frac{28}{31} \cdot t^2$
r_1	256 kbit/s	$(1 - t^2)$
r_2	2048 kbit/s	$\frac{3}{31} \cdot t^2$

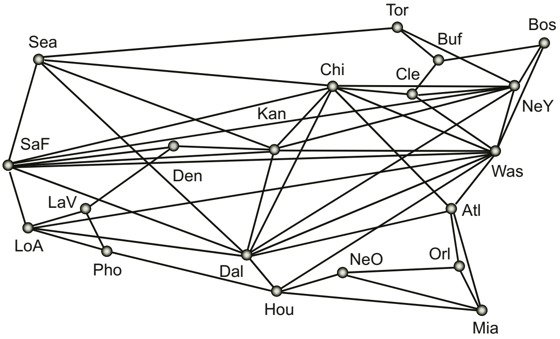


Fig. 1. Topology of the test network

of the request-type-specific inter-arrival and the mean of the call holding time determine the request-type-specific offered load $a(r_i)$. The overall load is $a = \sum_{0 \leq i < n_r} a(r_i)$. The random variable C_t indicates the requested rate in case of a flow arrival and the request size probability $P\{C_t = c(r_i)\}$ depends on the parameter $t \in [0, 1]$. The statistical properties of the request types are compiled in Table 1. They are chosen such that we get a constant mean of $E(C_t) = 256$ kbit/s and a coefficient of variation of $c_{var}(C_t) = 2.291 \cdot t$ that depends linearly on t.

Traffic Matrix. The network experiments in this paper are based on the KING [4] reference network given in Fig. 1. All network nodes are both ingress and egress routers. We scale the traffic matrix for the test network with the overall offered load a_{tot}. The generation of the traffic matrix is based on the population of the cities and their surroundings [2]). For two cities v and w with population sizes $\pi(v)$ and $\pi(w)$, the border-to-border (b2b) offered load $a(v, w)$ amounts to

$$a(v, w) = \begin{cases} \frac{a_{tot} \cdot \pi(v) \cdot \pi(w)}{\sum_{x,y \in \mathcal{V}, x \neq y} \pi(x) \cdot \pi(y)} & \text{for } v \neq w, \\ 0 & \text{for } v = w. \end{cases} \tag{1}$$

The average offered b2b load a_{b2b} specifies the overall offered load in the network $a_{tot} = \sum_{v,w \in \mathcal{V}, v \neq w} a(v, w) = |\mathcal{V}| \cdot (|\mathcal{V}| - 1) \cdot a_{b2b}$, where $\mathcal{V}$ is the set of all nodes in the network. We use shortest path routing in our experiments, which is the basis for most Interior Gateway Protocols (IGPs). Our reference populations are given in [2].

3.2 Capacity Dimensioning for AC: $M/G/n-0$

Capcity dimensioning for AC on a single link with a multi-rate Poisson flow model and the usage of effective bandwidths is the task of finding the capacity n of a multi-rate $M/G/n-0$ blocking system. The capacity n – the number of basic bandwidth units – must be chosen to accommodate sufficiently many flows in the network to fulfill the desired blocking probability $p_b = 10^{-3}$. The well-known Kaufman/Roberts algorithm presented in [14] computes the blocking probability for a given traffic mix and capacity. Our capacity dimensioning algorithm for AC performs a computational inversion of these formulae in an efficient way [2].

3.3 Capacity Dimensioning for CO: $M/G/\infty$

With CO, the number of flows in the system is not bounded. Therefore, dimensioning for CO on a single link with a multi-rate Poisson model can be done using a $M/G/\infty$ system. We calculate the equilibrium state probabilities of the system. The request types constitute the $k = n_r$ classes for which the k-dimensional state space is described by $X = \{x = (x_0, x_1, \ldots, x_{k-1}) \in \mathbb{N}_0^k\}$. With the class-specific arrival rate λ_i and the class-specific mean holding time $\frac{1}{\mu_i}$ the equilibrium state probabilities are

$$p(x) = \prod_{i=0}^{k-1} \frac{\rho_i^{x_i}}{x_i!} e^{-\rho_i} \tag{2}$$

with $\rho_i = \frac{\lambda_i}{\mu_i}$. The consideration of the request type rates $c(r_i)$ yields the required link capacity $c(x) = \sum_{i=0}^{k-1} c(r_i) \cdot x_i$ of state x. Thus, the required capacity C for the overprovisioned system is

$$C = \min_{C'}\{1 - \sum_{c(x) \leq C'} p(x) \leq p_v\}. \tag{3}$$

This is the smallest capacity such that the rates of the flows crossing the link exceed the link capacity at most with the desired QoS violation probability $p_v = 10^{-6}$. The calculation of the state probabilities is also known as the stochastic knapsack with infinite capacity [15]. Its solution was originally derived for the $M/M/\infty$ system but it is insensitive to the holding time distribution and holds for $M/G/\infty$ systems, too.

3.4 Extension to Networks

The algorithms for link capacity dimensioning must be extended to entire networks.

BBB NAC. If a flow wants to pass the network from node v to w, the BBB NAC checks the single budget $BBB(v, w)$. We dimension the size of the budget with the single link AC algorithm in such a way that $p_b = 10^{-3}$ is achieved. The capacity of a link within the network is the sum of all budgets crossing this link.

CO. For CO, the QoS violation probability on the complete path from source to destination must be at most $p_v = 10^{-6}$. The corresponding probabilities $p_v(l)$ on the individual links are clearly rather positively correlated. An upper bound for p_v on the path is given by $p_v(path) = 1 - \prod_{l \in path}(1 - p_v(l))$ where $l \in path$ denotes the links on the path and $p_v(l)$ is the QoS violation probability on the link. Now we can compute

$p_v(l) = 1 - \sqrt[len(path)]{1 - p_v}$ for a given link for every b2b relation and obtain the minimum of these values as the required QoS violation probability on this link. Based on $p_v(l)$ and the aggregate offered load $a(l)$ of all flows traversing link l we can dimension the capacity of each link l in the network.

4 Capacity Requirements for AC and CO

In this section, we compare the capacity requirements for AC and CO. We dimension the capacity for AC such that the blocking probability is $p_b = 10^{-3}$ under normal conditions. As CO cannot prevent overload situations, we dimension the capacity for CO in a very conservative manner such that the QoS violation probability is $p_v = 10^{-6}$. We concentrate first on a single link to understand the basic tradeoffs and then we extend our study to entire networks.

4.1 Single Link with Constant Load

First, we explain economy of scale as it is the key to understand the phenomena in our study. Then, we compare the capacity requirements for AC and CO for a constant load on a single link and consider the strength of the QoS violation by CO. Finally, we enhance the constant offered load scenarios by rare overload situations.

Economy of Scale. In Fig. 2 we dimensioned the required capacity on a single link for AC and a blocking probability of $p_b = 10^{-3}$. The required link capacity is almost proportional to the offered link load, at least for an offered load of 10^3 Erlang or more. The average resource utilization of that capacity by the offered traffic increases with the offered load and expresses the resource efficiency in a natural way. The fact that little offered load leads to low utilization and that large offered load leads to high utilization is a non-linear functional dependency and it is called economy of scale or multiplexing gain.

Figure 2 also shows that traffic with highly variable request sizes ($t = 1$) requires more capacity. In the following, we use only highly variable traffic ($t = 1$) due to the multi-rate nature of Internet traffic.

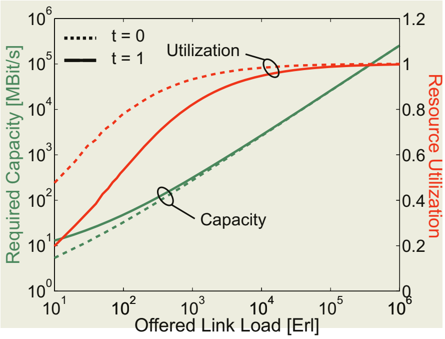


Fig. 2. Economy of Scale on a single link for AC

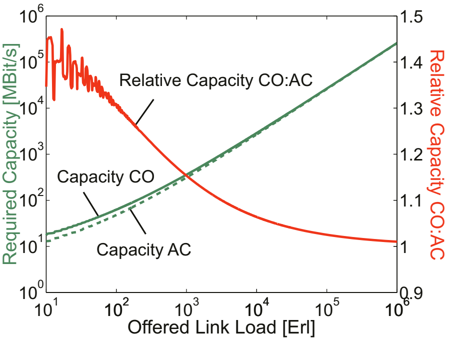


Fig. 3. Impact of offered load on capacity requirements for AC and CO

Comparison for Constant Offered Load. Figure 3 indicates the required capacity for AC and CO depending on the offered load. The ratio of both curves shows that they differ significantly only at low offered load. The oscillations here and in the following figures are due to the granularity limitation of the bandwidth and request size quantities. In particular, CO requires less than 5% additional capacity at 20^4 Erlang or more. The capacity for CO with $p_v = 10^{-6}$ equals approximately the capacity for AC with $p_b = 10^{-6}$. Due to economy of scale, this requires only slightly more capacity than AC with $p_b = 10^{-3}$ for large offered load [2].

Another concern is the degree to which QoS is violated. We can capture that by the lack of capacity in overload situations. The average lack of capacity over time is $E[L] = \sum_{c(x)>C, x\in X} (c(x) - C) \cdot p(x)$ where x is the state vector of flows in the system. Figure 4 illustrates this value in percent related to the provisioned capacity for CO. It is in the order of 10^{-6} for all considered scenarios because we have provisioned so much capacity that overload occurs very rarely with $p_v = 10^{-6}$. In case of overload situations, the lack of capacity is 6 orders of magnitude larger but it is not larger than 6%. The reason for that is the constant offered load in our experiment, which allows only small statistical oscillations but does not model occasional hot spots due to increased content attractiveness at certain locations.

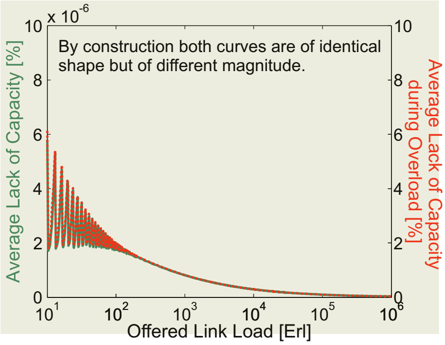


Fig. 4. Lack of capacity for CO as time average and conditioned on overload situations

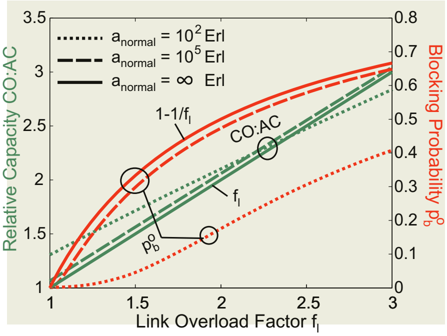


Fig. 5. Impact of rare overload on capacity requirements and blocking

Comparison for Rare Overload. Rare overload situations can occur due to hot spot scenarios caused by singular events. We capture this intuition by keeping the offered load at a normal level a_{normal} for a time fraction $\frac{364}{365}$ and increase it to an overload level of $a_{overload}$ for a short time fraction of $\frac{1}{365}$. In the following, we call the ratio $f_l = \frac{a_{overload}}{a_{normal}}$ the link overload factor. Note that the variability of the time series of offered load is still quite moderate with a coefficient of variation of 0.104 for an overload factor of $f_l = 3$. The capacity for AC is dimensioned based on a_{normal} since an increased blocking probability p_b can be tolerated for a short time interval whereas the capacity for CO is dimensioned based on $a_{overload}$ since CO cannot avoid congestion in severe overload situations. For very high offered load, a utilization of almost 100% can be achieved for AC. In this case, the ratio of the capacity requirements for CO and

AC scales with the overload factor f_l and the blocking probability p_b during overload situations scales with $1 - \frac{1}{f_l}$ which are both analytical values. Figure 5 shows these performance metrics for an offered load $a_{normal} = 10^2$ Erl and $a_{normal} = 10^5$ Erl. Regardless of the offered load, the ratio of the capacity requirements for CO and AC follows quite well the overload factor f_l while the blocking probability p_b^o depends also significantly on the offered load. The fact that the CO:AC capacity requirement curves cross is due to the stronger impact of multiplexing gain when the offered link load is low.

Figure 5 shows that the blocking probabilities p_b^o for $a = 10^{\{2,5\}}$ Erl are below the analytical value $1 - \frac{1}{f_l}$. In overload situations, 100% of the available bandwidth is used to transport traffic. If a high average utilization can be achieved under normal conditions, only relatively little extra capacity is available to accommodate extra traffic. Therefore, the blocking probability increases with offered load. Hence, QoS can be maintained with AC in overload situations and the effective blocking probability is significantly smaller than the simple analytical rule of thumb for a moderately aggregated traffic. However, the additional capacity for CO scales quite well with the assumed overload factor. These are the results from the analysis but there is another practical problem. The overload factor f_l is unknown and must be overestimated to guarantee QoS. This safety margin cannot be covered by our analysis but increases again the additional capacity requirements for CO.

4.2 Networks with Constant Load

We have studied the single link to understand the basic tradeoffs. If we proceed to entire networks, we have to take the impact of NAC into account which entails two different types of multiplexing gain.

A Traffic corresponding to a single b2b relationship (v, w) is carried within a single $BBB(v, w)$. Its offered load $a(v, w)$ determines the required capacity of that BBB and its utilization. Here, BBB NAC and CO can profit from economy of scale for budgets.

B Traffic corresponding to different b2b relationships is carried over a single link. The capacity requirement for the considered link is the sum of the capacities of the respective BBBs. With CO, in contrast, the required capacity of a link l is calculated on the basis of its overall offered traffic load $a(l)$ which is a larger aggregation level than the load pertaining only to a single budget. We call that economy of scale due to link sharing. It can be exploited by CO but not by BBB NAC.

Fig. 6 shows the capacity requirements for CO and AC and their ratio depending on the average offered b2b load a_{b2b}. CO requires significantly less capacity than AC for low offered load, and the capacity requirements are about the same for an offered load of 10^4 or more. We get this counterintuitive result because AC can exploit economy of scale only on the budget level (A) but not due to link sharing (B). The fact that p_v takes more stringent values than p_b plays obviously only a minor role in entire networks.

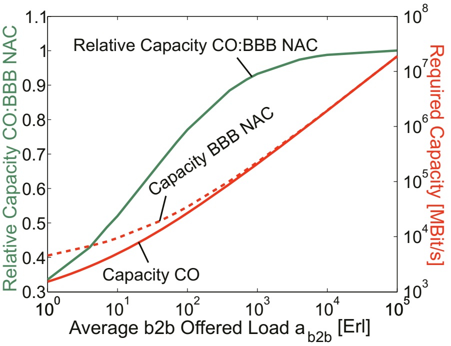


Fig. 6. Impact of offered load on the capacity requirements for CO and BBB NAC and their ratio in the KING testbed

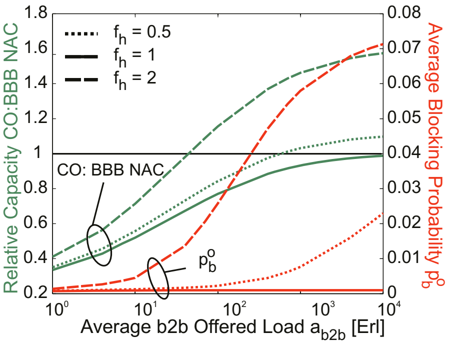


Fig. 7. Impact of offered load and hot spot factor on the ratio of capacity requirements for CO and BBB NAC, and blocking probabilities under overload conditions in the KING testbed

4.3 Networks with Rare Overload

The comparison of the capacity requirements is now enhanced by rare overload situations but it is still based on constant total offered load. AC can provide QoS also in such scenarios and a temporary increase in blocking can be accepted. Hence, capacity dimensioning is based on the normal traffic matrix. CO requires sufficient additional capacity for all overload conditions. Therefore, the link capacity must be provisioned for CO such that the maximum expected load can be carried for any overload scenario.

Overload Model for Networks. We model overload in entire networks quite conservatively by single hot spots whereby the overall offered load in the network does not increase, i.e., we change only the structure of the traffic matrix. We increase the traffic attraction of a single city v by a hot spot factor f_h, which is expressed by a modified population function

$$\pi^v_{overload}(w) = \begin{cases} \pi(w) & \text{if } w \neq v \\ f_h \cdot \pi(w) & \text{if } w = v \end{cases} . \tag{4}$$

For every potential single hot spot $v \in \mathcal{V}$, a traffic matrix is generated proportionally to $\pi^v_{overload}$ with the same overall offered load in the network.

Comparison of Capacity Requirements for Rare Over- and Underload. Figure 7 shows the relative network capacity CO:BBB NAC and the blocking probability p_b^o depending on the average offered b2b load a_{b2b} for $f_h \in \{0.5, 1, 2\}$. For a hot spot factor $f_h = 2$, CO already requires more capacity than BBB NAC for an offered b2b load $a_{b2b} = 70$ Erl and it needs 60% more capacity than BBB NAC for high offered load. During overload, the blocking probability is 7% – averaged over all b2b relationships – and it is at most 45.4% for a few b2b aggregates in some scenarios. Depending on the network operation policy, these values are well acceptable.

With $f_h = 0.5$, a single node looses global attractiveness, which means that the relative importance of all other cities is slightly increased. This causes a smooth shift of offered load in the traffic matrix from this city to other cities. It raises the capacity requirements for CO and the blocking probabilities for the BBB NAC only slightly in contrast to the previous experiment.

Figure 8 shows the impact of the hot spot factor f_h on the ratio of the capacity requirements for CO and BBB NAC as well as the corresponding average blocking probability p_b^o for an offered b2b load $a_{b2b} \in \{10^1, 10^3\}$ Erl. For the single link experiment, we could easily predict that the capacity requirements for CO scale with f_l. In case of a traffic shift due to increased attractiveness of a single node within an entire network, the effect of the hot spot factor f_h is significantly smaller. One reason is that BBB NAC can exploit economy of scale only within b2b budgets (cf. A above). This applies in particular for $a_{b2b} = 10$ and $a_{b2b} = 10^3$ Erl but not for clearly larger values of the offered b2b load. For $a_{b2b} = 10^3$ Erl and $f_h = 3$, the additional capacity requirements are about 200% for the single link experiment while they amount to only 150% for the network experiment. Another reason is the following. In a situation with an increased attractiveness of node v, the links leaving from and leading to v require most additional capacity. But they carry also transit traffic whose rate is rather slightly decreased by v's increased attractiveness. Hence, the increase of the capacity requirements of those links depends on a mixture of slightly decreased transit traffic rates and significantly increased rates for hot spot traffic. Therefore, their additional capacity requirements are smaller for a given hot spot factor f_h than the additional capacity requirement of a single link with a link overload factor f_l.

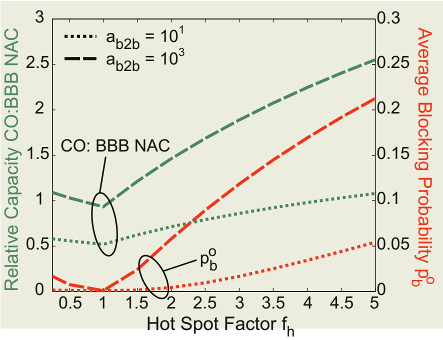


Fig. 8. Impact of the hot spot factor f_h on capacity requirements and blocking

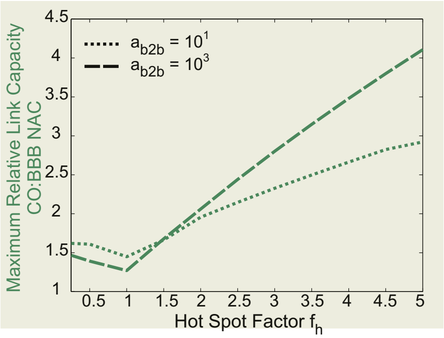


Fig. 9. Impact of the overload on the maximum additional relative link capacity requirements

Figure 9 shows the maximum values for the relative capacity requirements of CO compared to BBB NAC on all single links within the network. These maximum relative capacities are significantly larger than for the entire network. For example, the network requires only 150% more capacity for CO than for AC in case of a hot spot factor of 400%, but some links require 300% more capacity. Hence, the additional capacity varies significantly among the links and the exact amount depends on the network topology, the traffic matrix, and the routing. Therefore, determining the appropriate degree of overdimensioning for individual links in a network is a non-trivial task for which our analysis can be useful. It may be applied, e.g., to provision Differentiated Services networks [16], the base architecture for the future Internet, which will be a multi-service network with high and low priority traffic and suitable scheduling mechanisms. If the fraction of high priority traffic is low, bandwidth overprovisioning can be done for high priority traffic and the required excess capacity may be used under normal conditions to transport low priority traffic. In case of overload in the high priority traffic class, low priority traffic is swamped out.

Finally, we would like to point out that all these results were obtained by a mere traffic shift while the overall offered load in the network has been kept constant.

5 Discussion and Conclusion

Our results show that capacity overprovisioning (CO) requires only 5% more capacity on a single link than border-to-border (b2b) budget (BBB) based network admission control (AC) if we consider constant offered load. As this assumption is not realistic, we took temporary overload of up to 300% of the normal traffic into account. In this case, AC can guarantee QoS for admitted flows at constant capacity at the expense of higher blocking probability, whereas CO needs 200% more capacity. We extended this experiment to entire networks to better motivate the overload scenario. We assumed the traffic matrix to be proportional to the population of the catchment area of a router. To solicit overload, we kept the overall traffic in the network constant, increased the attractiveness of a single city by up to 400% by increasing the population by that factor, and performed that experiment for every city. This models realistic temporary hot spots without increasing the overall traffic volume. Here, CO requires 60% (150%) more capacity than AC for a hot spot factor of 100% (400%) while AC reacts with a blocking probability of 7% (25%).

Based on our temporary hot spot model, we could prove considerable bandwidth savings by AC compared to CO. AC, however, requires a substantial amount of signalling, coordination and interoperation that is not yet implemented in most networks. An economic assessment must take this into account.

We showed that the required degree of overdimensioning varies among the links within a network and, therefore, our analysis can be useful to determine the appropriate additional capacity for individual links for a given networking scenario with suitable assumptions regarding overload. This approach can be of particular interest for Differentiated Services networks [16].

Currently, we are working on a comparison between CO and link-by-link (LB) NAC. LB NAC is wider spread than BBB NAC and its resource utilization differs significantly. In addition, we intend to integrate resilience aspects into our work.

Acknowledgment

The authors would like to thank Prof. Tran-Gia for the stimulating environment which was a prerequisite for that work.

References

1. Odlyzko, A.: Data Networks are Lightly Utilized, and will Stay that Way. The Review of Network Economics **2** (2003)
2. Menth, M.: Efficient Admission Control and Routing in Resilient Communication Networks. PhD thesis, University of Würzburg, Faculty of Computer Science, Am Hubland (2004)
3. Kelly, F.P.: Notes on Effective Bandwidths. In: Stochastic Networks: Theory and Applications. Volume 4. Oxford University Press (1996) 141 – 168
4. Hoogendoorn, C., Schrodi, K., Huber, M., Winkler, C., Charzinski, J.: Towards Carrier-Grade Next Generation Networks. In: ICCT, Beijing, China (2003)
5. Fraleigh, C., Tobagi, F., Diot, C.: Provisioning IP Backbone Networks to Support Latency Sensitive Traffic. IEEE Infocom 2003, San Francisco, USA (2003)
6. Kilkki, J., Norros, I.: Testing the Gaussian Approximation of Aggregate Traffic. In: Internet Measurement Workshop, Marseille, France (2002)
7. Van Hoey, G., De Vleeschauwer, D.: Benefit of Admission Control in Aggregation Network Dimensioning for Video Services. Networking 2004, Athens, Greece (2004)
8. Van den Berg, H., et al.: QoS-aware Bandwidth Provisioning for IP Network Links. Technical Report PNA-E0406, Centrum voor Wiskunde en Informatica, The Netherlands (2004)
9. Papagiannaki, D., Taft, N., Zhang, Z., Diot, C.: Long-Term Forecasting of Internet Backbone Traffic: Observations and Initial Models. In: IEEE Infocom, San Francisco, USA (2003)
10. Tuan Tran, H., Ziegler, T.: Adaptive Bandwidth Provisioning with Explicit Respect to QoS Requirements. Quality of Future Internet Services (QoFIS), Sweden (2003)
11. Paxson, V., Floyd, S.: Wide-Are Traffic: The Failure of Poisson Modelling. IEEE/ACM Transactions on Networking (1995)
12. Roberts, J.W.: Traffic Theory and the Internet. IEEE Communications Magazine **1** (2001) 94–99
13. Dinh, T., Sonkoly, B., Molnár, S.: Fractal Analysis and Modeling of VoIP Traffic. In: Proceedings of Networks 2004, Vienna, Austria (2004) 123 – 130
14. Roberts, J., Mocci, U., Virtamo, J.: Broadband Network Teletraffic - Final Report of Action COST 242. Springer, Berlin, Heidelberg (1996)
15. Ross, K.W., Tsang, D.H.K.: The Stochastik Knapsack Problem. IEEE Transactions on Communications **37** (1989) 740–747
16. Xiao, X., Ni, L.M.: Internet QoS: A Big Picture. IEEE Network Magazine **13** (1999) 8–18

RIDA: Robust Intrusion Detection in Ad Hoc Networks

Dhanant Subhadrabandhu[1], Saswati Sarkar[1,⋆], and Farooq Anjum[2]

[1] Electrical and Systems Engineering Department,
University of Pennsylvania
[2] Telcordia Technologies

Abstract. We focus on detecting intrusions in wireless ad hoc networks using the misuse detection technique. We allow for detection modules that periodically fail to detect attacks and also generate false positives. Combining theories of hypothesis testing and approximation algorithms, we develop a framework to counter different threats while minimizing the resource consumption. We obtain computationally simple optimal rules for aggregating and thereby minimizing the errors in the decisions of the nodes executing the intrusion detection software (IDS) modules. But, we show that the selection of the optimal set of nodes for executing the IDS is an NP-hard problem. We present a polynomial complexity selection algorithm that attains a guaranteeable approximation bound. We also modify this algorithm to allow for seamless operation in time varying topologies, and evaluate the efficacy of the approximation algorithm and its modifications using simulation. We identify a selection algorithm that attains a good balance between performance and complexity for attaining robust intrusion detection in ad hoc networks.

1 Introduction

Ad hoc networks provide the only means of electronic communication in areas where establishing infrastructure like base stations is either impossible or not cost-effective. Examples include disaster recovery operations, battlefields, communication in remote terrains (e.g., reservations, rural areas), events like superbowl matches, etc. Ad hoc networks do not need infrastructure because users perform some communication tasks like relaying packets. But, if the network is to provide any quality of service (QoS) guarantee it can utilize the users but can not solely rely on them. This is because users may be available for short durations only. The QoS guarantees can however be provided if some easily deployable low complexity system nodes e.g., static and mobile access points are available. These nodes together with users who are trusted by the network and are in the network most of the time can be relied upon for relaying packets, discovering routes, securing the communication, etc. Thus, there are two sets of nodes: users who only communicate using the network but do not perform system tasks (outsider nodes), and static and mobile access points[⋆⋆] and users that communicate and perform network tasks

[⋆] The contribution of S. Sarkar was in part supported by the National Science Foundation under grants ANI-0106984, NCR-0238340 and CNS-0435306.

[⋆⋆] These access points operate in the ad hoc mode and not in the infrastructure mode of 802.11 networks.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1069–1082, 2005.

(insider nodes). As an example, consider a wireless network in a university. Universities have static access points deployed in several places. Mobile access points carried by personnel and vehicles can provide additional coverage as required, e.g., in areas of heavy network traffic like games, concerts, etc. This motivates the utilization of ad hoc networks in universities. In such ad hoc networks, the terminals of the university employees can also perform system tasks as required. The access points and the employees' terminals constitute insider nodes. Students as well as other visitors constitute outsider nodes. As another example, a disaster recovery team can use ad hoc networks to provide services like email, news, audio/video applications etc. in an area where communication infrastructure has been damaged due to a natural disaster or terrorist activity. The insider nodes are small access points on buildings and mobile terminals carried by the personnel. The outsider nodes are civilians who communicate with each other using the network. We postulate that ad hoc networks deployed in near future will match this description. We address security threats in these.

These networks will be used by a diverse user population, e.g., civilians in disaster hit areas, students in universities etc., which increases the security risks. One such risk is a user who subverts the functioning of the network by causing undesirable events. Such users are considered as intruders and the events as intrusions. Examples of intrusions are attacks such as TCP SYN flood*, Land Exploit**, SSPing*** etc. [1] [2]. These intrusions leverage system vulnerabilities. There are two ways to prevent such intrusions. One way is to remove the vulnerabilities from the system such as by designing resistant protocols like SCTP [3] to resist TCP SYN flood attacks, patching the operating systems, etc. But, this may not be possible due to various reasons such as poor design [4], limited use of efficient technical solutions (e.g., SCTP is rarely used due to large scale deployment of TCP), different devices having different capabilities, inefficient configuration (e.g., users do not change default security settings or apply patches), etc. The second approach, which is complimentary to the first, is to detect attempts to leverage the vulnerabilities and stop such attempts from succeeding. In this paper, we focus on the second approach which is referred to as intrusion detection.

Intrusion detection has been extensively investigated for wireline networks [5], [6]. But techniques geared towards wireline networks would not suffice in an ad hoc network due to the ease of listening to wireless transmissions, lack of fixed infrastructure, etc. [7]. For example, several detection strategies in wireline networks are based on the presence of a small number of gateways that route and therefore monitor all traffic. But, ad hoc networks typically do not have such choke points and even if such choke points exist, their locations continuously change due to mobility. Thus, designing the optimal selection strategy is more complex in ad hoc networks. Also, intrusion may be detected in wireline networks by detecting anomaly, i.e., by comparing the current system behavior with that in absence of intrusion. In ad hoc networks, however, normal behavior can not be accurately characterized, e.g., a node may transmit false updates since the routing protocol is slow to converge and not because it is malicious. Further,

* The attacker opens a large number of half-open TCP connections.

** The attacker sends a TCP SYN packet with the same target and source address.

*** The attacker sends a series of highly fragmented, oversized ICMP packets.

unlike in wireline networks, nodes in an ad hoc network have limited energy. Hence, only computationally simple, energy-efficient detection strategies can be used. The detection algorithms must also be distributed as communication with a central computing unit will consume significant energy. Finally, the detection algorithms must seamlessly adapt to topological changes due to mobility.

A detection strategy specifically suitable for ad hoc networks is that of misuse detection that relies on the use of known patterns of unauthorized behavior [8]. More specifically, this technique detects intrusion when the transmitted traffic contains abnormal packets which serve as "signatures" of attacks. For example, the signature of SSPing attack is a series of highly fragmented, oversized ICMP packets. A SYN packet with the same source and destination address indicates a land exploit attack [1]. The advantage of this technique in ad hoc networks is that it does not require characterization of normal behavior. This technique can not however detect attacks whose signatures are unknown.

The misuse detection technique requires some nodes to capture (sniff) and analyze the network traffic. Therefore, a prerequisite for deploying misuse detection in ad hoc networks is to determine which nodes should execute the sniffing and analysis software modules which we refer to as the intrusion detection software (IDS) modules. Previous works have considered this problem assuming that the insider nodes that analyze the traffic detect malicious packets without any failure [8], [9], [10]. But some insiders periodically stop functioning because of operational failure and low residual energy and would not detect attacks during those intervals. Insider nodes may also erroneously conclude intrusion when there is none. It is difficult to quickly detect which insider nodes have failed, and attacks may be deliberately launched before the IDS can be activated in other nodes.

We focus on the misuse based detection problem in ad hoc networks while allowing for erroneous decisions made by insider nodes executing the IDS modules. We describe our system model in Section 2. Combining theories of hypothesis testing and approximation algorithms, we develop a framework to counter different threats while consuming the minimum possible resource (Section 3). We obtain computationally simple optimal rules for aggregating and thereby minimizing the errors in the decisions of the nodes detecting the intrusion. But, we prove that optimally selecting the nodes for sniffing and analyzing packets is NP-hard. Then, we present an approximation algorithm APPROX that attains a guaranteeable approximation bound in polynomial complexity. We also modify APPROX to ensure seamless operation in time varying topologies. Using simulations, we evaluate the detection costs and security risks of different algorithms and identify one selection algorithm MUN that attains a good balance between performance and complexity for attaining robust intrusion detection in ad hoc networks (Section 4). Refer to technical report [11] for proofs. Details related to implementation of the proposed schemes are beyond the scope of this paper.

2 System Model

A network consists of two types of nodes: insider nodes and outsider nodes. An outsider node may launch attacks on another outsider or an insider node. Therefore, any outsider node that sends traffic is potentially malicious, and is referred to as an intruder. There may be multiple intruders who may use any set of paths for transferring their packets.

The number and the locations of the outsider nodes and their destinations are not known to the network, and vary with time. We assume that each attack consists of one packet, e.g., a land exploit attack [1] consists of one packet. Packets constituting attacks are denoted as *bad* packets. A packet not constituting an attack is denoted as a *good* packet.

We represent a wireless network by an undirected graph $G(V, E)$. Here, $V = \{1, \ldots, N\}$ consists of the insider nodes and E is the set of edges between the insider nodes. There exists an undirected edge between any two insider nodes which are in each others' transmission range.

Definition 1. *A neighborhood N_i of an insider node i is the set of insider nodes that are within i's transmission range. An insider node i covers every insider node in its neighborhood.*

Thus, an insider node is always its own neighbor and covers itself.

Insider nodes execute the IDS modules that employ misuse-based detection strategy so as to detect bad packets while in transit between the intruder and the destination. Some insider nodes may not have the capability to execute the IDS. Thus, insider nodes are of two types: (a) *IDS capable* and (b) *IDS incapable*. Also, different IDS capable insider nodes e.g., PDAs, laptops, access points etc. consume different amount of resources to execute the IDS, since they have different residual energy and computational capability. An IDS capable insider node i has weight w_i that represents its resource consumption when it executes the IDS. Depending on the system policy, some but not all the IDS capable insider nodes will execute the IDS - these are denoted as *IDS active*. The set of IDS active nodes may dynamically change depending on available bandwidth, computational resources, energy and the current topology.

An IDS active insider node operates in promiscuous mode, i.e., receives any packet that is transmitted by any of its neighbors. Several authors e.g. [8],[12], assume operation in promiscuous mode given its advantages (e.g., neighborhood monitoring).

An IDS active insider node may sporadically fail to detect bad packets and report good packets as bad. A node may not detect bad packets during power saving operations, or if the attack has been designed to evade its IDS module [13],or if the IDS modules on the node are out of date, or if the attacker successfully launches a denial of service (DOS) attack on the node, or if it does not receive the packets due to collisions[†], poor transmission quality in wireless links, etc. Depending on the signature matching techniques used an insider may also report a good packet as bad. Bloom filters [14] are examples of such techniques. Bloom filters consider a packet or packet fragments to be suspicious whenever a hash function of the packet or packet fragments match that of an attack signature. Only suspicious packets are analyzed more thoroughly. This technique can be implemented using hardware and therefore signatures can be matched at line speed. But, this technique also results in false positives. The false positive rate can sometimes be non-negligible, e.g., sometimes 10% of the good packets have been reported to be

[†] Some collisions happen only due to promiscuous operation. Consider an ad hoc network with 3 insider nodes A, B, C. All nodes are IDS capable. Let both A and C be B's neighbors. But, A and C are not each others neighbors. Let B execute the IDS. If A and C simultaneously relay bad packets to outsider nodes, the packets collide at B. Since however the intruders transmit bad packets only rarely, such collisions of bad packets are rare.

suspicious [14]. Now when an insider node is overloaded or does not have enough resources it may report suspicious packets as bad without conducting a thorough analysis. Otherwise, if the node chooses to completely analyze suspicious packets it may have to drop several incoming packets which are more likely to be good.

We assume that every IDS active insider considers a bad packet to be good with probability p and a good packet to be bad with probability q. We focus on the efficacy of detection schemes given such a failure model. We do not consider the threat of compromise whereby insider nodes are under the control of the intruders. We assume that the values of p and q are known. This knowledge can be attained through online measurements. Due to space constraints, we do not describe such measurement techniques. Finally, we assume that the p and q are the same for every insider. Future research will be directed towards generalizing the framework for different values of p and q for different insiders.

There exist different strategies for selecting IDS active nodes. An obvious IDS placement strategy (host intrusion detection or HID) [15] is to execute the IDS at only the destinations of the sessions. Here, a node executes the IDS at its application layer, and can therefore analyze only the packets it receives as destination, and not those it relays. Another strategy is to use the network intrusion detection (NID) technique [15] where the IDS is executed on some selected insider nodes, which may be relays or end-hosts. Here, a node executes the IDS at its network layer, and can therefore analyze both the packets it relays and receives as destination. We consider NID in the rest of this paper.

The challenge in deploying NID is to appropriately select the IDS active nodes. A straightforward strategy is to execute the IDS on every insider node. Thus more bad packets will be examined. But, this clearly consumes significant energy and requires substantial computation. On the other hand, if the IDS are executed in very few nodes, then the resource consumption decreases but some bad packets may escape inspection. The security risk in each case will depend on the nature of the imperfections. In the first case, the detection rate will be higher as each packet is examined by higher number of insiders. But, since more insiders examine each packet, more false positives may be generated and the number of good packets dropped may be higher (if a node drops packets marked as bad). The challenge now is to select the IDS active nodes such that even with these imperfections, the required detection and drop rates are achieved while limiting the resource consumption.

We assume that either a packet is not encrypted or only the transport layer payload of a packet is encrypted. This is the case with several protocols like PGP, SRTP, HTTPS etc. Then, IDS modules can detect attacks at the transport and lower layers, e.g., ping-of-death, TCP SYN flood, etc. without any knowledge of the encryption schemes. We do not consider link layer and network layer encryption protocols like IPSec since these protocols are typically used in enterprise environments which is beyond the scope of this paper. This is a standard assumption in several papers that investigate NID [15] .

3 Algorithms for Robust Intrusion Detection

We first consider the detection goals. Clearly, a detection goal would be to maximize the probability of detecting bad packets. But, since an IDS active insider reports some good packets as bad, we also need to minimize the probability of reporting a good packet as bad (*"false positive"*). It may not be possible to attain both goals simultaneously. This happens when increasing the detection rate involves increasing the false positive rate. Thus, our goal is to select the IDS active insiders among the IDS capable insiders so as to minimize the resource consumption subject to attaining a *risk* which is below an acceptable value. We quantify the risk as a weighted sum of the probability P_M that a bad packet is not detected (*missed detection*) and the probability P_F of false positive, $y_M P_M + y_F P_F$, where y_F, y_M are pre-specified weights. The resource consumption is the sum of the weights of all IDS active insiders.

The risk needs to be maintained below an acceptable value without any knowledge of the intruders, targets and the paths between them. This can be done only when the IDS active insiders are selected so that every packet is analyzed and the results of the analysis intelligently combined to reduce the errors in the decisions. If every IDS active insider could decide without any error ($p = q = 0$), then most of the packets would be analyzed if the IDS active insiders are selected so that every insider is a neighbor of at least one IDS active insider. In this case, every packet transmitted by an insider would be analyzed at least once and every insider detects whether a packet is bad without any error. Only the packets transmitted directly from an intruder to its target may not be analyzed, but the percentage of such packets is small [9]. This suggests that when insiders may decide erroneously, the IDS active insiders must be selected so that every insider is a neighbor of at least k IDS active insiders where $k > 1$. Now, the coverage redundancy may allow insiders to correct errors in decisions (Figure 1)[‡].

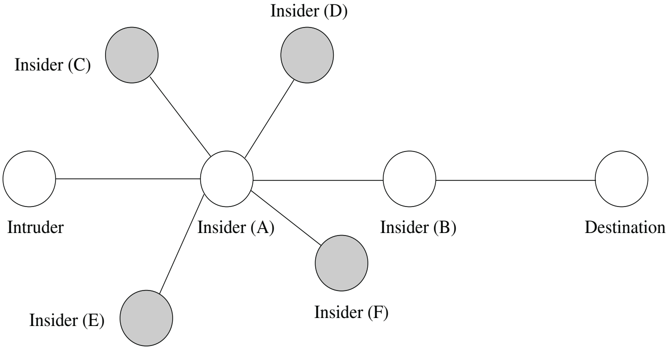


Fig. 1. This figure illustrates the coverage redundancy of an insider. The intruder attacks the destination. Insider nodes A and B relay the insider's packets to the destination. Node A is covered by IDS active insiders (C, D, E, F). When A relays a packet, C, D, E, F receive the packet in promiscuous mode. If any of these detect the packet to be bad, it reports its diagnosis to A. Based on the reports, A determines whether the packet is bad

[‡] Using Figure 1, we describe a preliminary recovery protocol. If based on its neighbors' inputs, A determines a packet it relays to be bad, it reports its diagnosis to the next relay B. B holds each packet for some time before relaying it to the destination. If A reports the packet to be bad, B drops it. If A does not receive any report from B in the pre-determined interval, it delivers the packet to the destination. Details regarding the recovery protocol is beyond the scope of this paper.

There are two questions that we now need to answer: (a) how does an insider optimally decide whether a packet is bad and (b) what is the optimal value of k. The issues clearly depend on each other. First consider the difficulties in determining (a). Due to the coverage redundancy several insiders may analyze a packet, and they may decide differently whether the packet is bad. The different decisions must be combined to determine whether the packet is indeed bad. For example, in Figure 1, if C and D detect a packet relayed by A to be bad and E and F determine otherwise, A needs to decide whether it should report an attack. The challenge is to aggregate the neighbors' decisions so as to minimize the expected risk. We attain this objective by developing an optimal aggregation scheme based on hypothesis testing framework. Now consider the difficulties in determining (b). We prove that under the optimal aggregation scheme the expected risk decreases with increase in k. But, the resource consumption also increases with increase in k. The challenge therefore is to select the minimum k that attains a tolerable risk when each insider optimally aggregates its neighbors' decisions. Once k is determined, we need to select the IDS active insiders so as to minimize the resource consumption or the total weight of the IDS active insiders subject to ensuring that every insider is covered by at least k IDS active insiders.

We determine the optimal k, the algorithms for aggregating the decisions and the optimal selection of IDS active insiders under the assumption that only one insider relays each packet. Using simulations, we investigate the performance of different algorithms when each packet is relayed by an arbitrary number of insiders.

We first obtain the optimal aggregation scheme at an insider i which has k IDS active neighbors. Consider a packet A selected with uniform probability among all packets relayed by i. Then, A is a good packet w.p. π_G, where π_G is the probability that an arbitrary packet is good [§] Now, the expected conditional risk for A is $H_i(k) = y_M P_{iM} + y_F P_{iF}$, where P_{iM} and P_{iF} are the respective probabilities of missed detection and false positive at i, which depend on i's aggregation scheme. Now, given k, i's optimal aggregation scheme is one that minimizes $H_i(k)$, and the minimum value of $H_i(k)$ is referred to as $H(k)$.

Theorem 1. *Given k, an insider attains $H(k)$ if it uses the following aggregation scheme. Let threshold $T = \left\lceil \frac{\ln \frac{y_F \pi_G}{y_M(1-\pi_G)} + k \ln \frac{1-q}{p}}{\ln((1-p)(1-q)/pq)} \right\rceil$. When $p + q < 1$,[¶] the insider node decides that a packet is bad if and only if T or more of its IDS active neighbors inform that the packet is bad. When $p + q \geq 1$, the insider node decides that a packet is bad if and only if fewer than T of its IDS active neighbors inform that the packet is bad.*

Theorem 2. *When $p + q < 1$, $H(k) = y_F \pi_G \sum_{i=T}^{k} \binom{k}{i} q^i (1-q)^{k-i} + y_M(1 - \pi_G) \sum_{i=0}^{T-1} \binom{k}{i} p^{k-i} (1-p)^i$. When $p + q \geq 1$, $H(k) = y_F \pi_G \sum_{i=0}^{T-1} \binom{k}{i} q^i (1-q)^{k-i} + y_M(1 - \pi_G) \sum_{i=T}^{k} \binom{k}{i} p^{k-i} (1-p)^i$.*

We now present the intuition behind the results.

[§] We assume that each insider knows π_G. Statistical techniques, e.g., Baye's minimax framework [16] can be used to develop optimal aggregation rules when this is not the case. Again, we can not describe such techniques due to lack of space.

[¶] Note that p and q are probabilities associated with different packets. Thus, $p + q$ can exceed 1.

When $p + q < 1$, the probability of error is small. So, a large number of insiders are likely to report a packet as bad only when the packet is bad. Thus, an insider decides the packet is bad only when many of its IDS active neighbors report it as bad. When $p + q \geq 1$, probability of error is high. So, if a packet is bad, many insiders would report it as good. Thus, the previous policy is reversed. We computed the aggregation thresholds and the minimum risk $H(k)$ for an IDS active insider using the theory of hypothesis testing [11],[16]. Finally, note that the aggregation is optimum for an insider irrespective of whether it is IDS active. Since each insider is also its own neighbor, an IDS active insider executes the above aggregation rule considering both its and its other neighbors' analysis of each packet.

We now obtain the optimum value of k and the optimum set of IDS active insiders, D. First, we obtain the following result.

Corollary 1. *As k increases, $H(k)$ decreases.*

Intuitively, an insider can make better decisions, if it has more information, i.e., if it hears from more neighbors.

Let the tolerable accepted risk be γ. Since only one insider relays each packet, each packet has an expected risk of $H(k)$. Let $k_{\min} = \arg\min_k \{H(k) \leq \gamma\}$. The detection goal of attaining the tolerable expected risk subject to minimizing the total weight of the IDS active insiders is now satisfied if the IDS active insiders are selected so as to minimize their total weight subject to ensuring that each insider has at least $k_{\min}$ IDS active neighbors. We now discuss how to select the IDS active insiders so as to attain the above goal. We first introduce some terminologies.

Definition 2. *A k-multicover in G is a set of insiders such that every insider in G is covered by k insiders in the set.*

Definition 3. *An IDS capable k-multicover is a k-multicover such that its members are IDS capable.*

Definition 4. *An IDS capable minimum weighted k-multicover is an IDS capable k-multicover with minimum total weight among all IDS capable k-multicovers.*

Clearly, the set of IDS active insiders need to be an IDS capable minimum weighted k-multicover in G. It is well-known that computing a minimum weighted k-multicover is an NP-hard problem [17]. This motivates the following lemma.

Lemma 1. *Optimally selecting the IDS active insiders is an NP-hard problem.*

There may not be any k-multicover in G, e.g., when an insider node in G is covered by at most $k - 1$ insider nodes. In this case, we have to opt for maximum possible coverage, which we explain later.

We next present an approximation algorithm (APPROX) for selecting the IDS active insiders. This has been obtained by modifying a greedy algorithm in [18] to accommodate IDS incapable insiders and still provide performance guarantees.

We introduce some terminologies required to describe the algorithm. The algorithm progressively adds IDS active insiders in a set D, which is initially empty. An insider in G is "satisfied" if it is covered by k or more insiders in D, and unsatisfied otherwise. Initially, every insider is unsatisfied. Let $US(u)$ be the number of unsatisfied neighbors of an insider node u in G. Let I be the set of IDS capable insiders in G. When the algorithm terminates, if there exists at least one IDS capable k-multicover in G, all insiders are satisfied and D is a k-multicover.

1. Let every insider be unsatisfied and $D = \phi$.
2. Let u be an IDS capable insider such that $w_u/US(u) = \min_{v \in I \setminus D} w_v/US(v)$.
3. $D = D \cup \{u\}$.
4. If any of u's neighbors v has k neighbors in D, v's status changes to satisfied.
5. Terminate if all the neighbors of the insiders in $I \setminus D$ are satisfied. Otherwise, go to step 2.

Note that if an insider u is not added in D, then insiders must be added to cover $US(u)$ insiders. Thus, $US(u)$ reflects the reduction in the coverage requirement brought about by adding u in D. The algorithm requires $|I|$ or less iterations, and has complexity $O(|I|)$.

Theorem 3. *Let G have at least one IDS capable k-multicover. Then, D is an IDS capable k-multicover with weight (resource consumption) at most $\sum_{i=1}^{d+1}(1/i)$ times that of the weight of an IDS capable minimum weight k-multicover in G, where d is the maximum degree of an insider node in G.*

Theorem 4. *Let G have no IDS capable k-multicover. An insider either has at least k IDS active neighbors or all its IDS capable neighbors are IDS active.*

It is in this sense that D maximizes coverage in G, when G has no IDS capable k-multicover. Each insider can decide whether to execute the IDS in a distributed manner, if it learns $\min_{v \in I \setminus D} w_v/US(v)$ by exchanging messages with its neighbors. This can be attained if a pre-determined root node broadcasts a packet with a value 0 and every IDS capable insider u which is IDS inactive and has $US(u) > 0$ updates the content of the message with the minimum of the current content and $w_u/US(u)$, and forwards it along the broadcast tree. The leaf nodes of the tree return the packet towards the root. If an insider executes the IDS, it informs its neighbors, and they change their status to satisfied if required. Each insider also informs its neighbors about whether it is satisfied. The insiders learn about the updated $\min_{v \in I \setminus D} w_v/US(v)$ by another broadcast. The root node does not broadcast when it receives a return packet with content 0. Clearly, at most $|I|$ broadcasts are necessary if the insiders do not move.

This algorithm for selecting the IDS active insiders is oblivious to the position of the outsiders, and is therefore not affected by their movements. But, the IDS active set must be recomputed each time an insider node's neighborhood changes due to its or its neighbors' movements. The computations and the related message exchanges consume significant resources particularly when they are executed frequently, i.e., when the insider nodes move rapidly. We now present computationally simple algorithms that do not require any re-computation with movement of either insider or outsider nodes, and require only limited message exchange when insider nodes move. The disadvantage is that we have not been able to prove any approximation bound for any of these algorithms. We evaluate them using simulation.

First we modify APPROX. Now, an IDS capable insider node u periodically exchanges messages with its neighbors to learn $w_v/US(v)$ for each neighbor v. After obtaining this information, u executes the IDS if and only if $US(u) > 0$ and $w_u/US(u)$ is the minimum in u's neighborhood. Then, u informs its neighbors about whether it is IDS active. This modified version is referred to as MUN (Maximum Unsatisfied Neighbors). Clearly, if the insider nodes do not move, the set of IDS active insiders stabilizes after some time. When insider nodes move, the decisions are updated in their neighborhoods, but the updates involve simple computations and limited message exchange. Since the decisions now depend on local instead of global minimums, we could not prove any approximation bound. Nevertheless, extensive simulations show that MUN and APPROX have similar performance.

Now, we consider a naive algorithm, Random Placement (RP), in which every IDS capable insider node executes the IDS with a probability which can be selected so as to regulate the resource consumed and the expected risk. For example, if this probability is high, then a large number of insiders are IDS active. Thus, the scheme consumes a lot of resource but the expected risk is likely to be low. This algorithm requires no message exchange. In Section 4, we compare the performances of APPROX, MUN and RP, and determine when each may be deployed.

4 Performance Evaluation

Using ns2-simulations, we compare the performance of the approximation algorithm, APPROX, and the computationally simple heuristics MUN and RP for selecting the IDS active insiders. We consider the optimal aggregation rule in each case. The simulations allow us to investigate the effect of the factors we did not consider in the analysis such as arbitrary number of hops between the intruder and the target, mobile insiders etc. We also evaluate the benefits of intelligently selecting the IDS active nodes, and accordingly decide the appropriate algorithm for any desired tradeoff between risks and resource consumption. Sample computations suggest that the approximate selection algorithm, APPROX, closely approximates the optimal solution, and the performance difference is generally much less than the upper bound presented in Theorem 3; we do not include these comparisons due to space constraint [11].

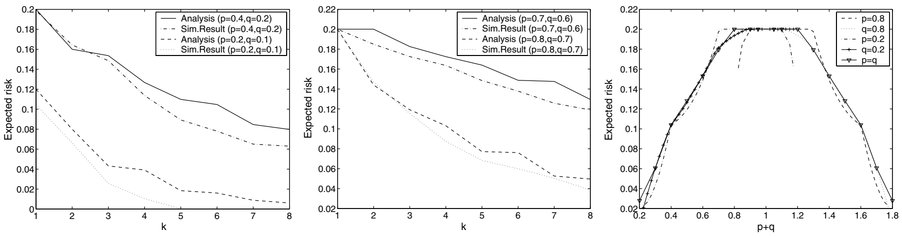
We consider networks with different types of node mobility, e.g., mobile intruders and static insiders, mobile intruders and mobile insider nodes etc. Each mobile node moves as per the random way point model with a maximum speed of 20 m/s and pause time 10 sec. For each combination, we measure averages over 300 different topologies. Each topology consists of a single intruder, a single target and 100 insider nodes. The insider nodes are uniformly distributed in a square of side 600m. Every insider is IDS capable and has unit weight. Every node has transmission radius 150m. Each attack consists of a single packet, and 20% of the packets transmitted by the intruder are bad (i.e., constitute attacks). Thus $\pi_G = 0.8$. We assume $y_M = y_F = 1$. For each topology, we measure the expected risk as the sum of the bad packets that are not detected and the good packets that are reported as bad divided by the total number of packets. We measure the detection cost as the total number of IDS active insiders.

In figure 2(a) and (b) we compare the analytical results with the simulation results. We plot $H(k)$ (obtained from Theorem 2) and the expected risk measured in the simulations as a function of k. We consider networks where every packet is relayed by a single insider node. For each k, in each topology, we select the IDS active insiders using the APPROX algorithm. Then, we select an insider uniformly among all the insiders and measure the risk when it relays the packets. The intruder and its target are selected within the transmission range of this insider. Here, we assume that all nodes are static. We consider different values of p, q: (i) two sets of p, q with $p+q < 1$ (figure 2(a)), and ii) two sets of p, q with $p+q > 1$ (figure 2(b)). Since the number of insiders is 100, in most trials each insider has fewer than 9 neighbors. Thus, we could vary k only until 8. We see that the expected risk measured in simulations is close to, but lower than $H(k)$. The difference can be explained as follows. APPROX only ensures that every insider has at least k IDS active neighbors. Thus, some insiders have more than k IDS active insiders. But, $H(k)$ is the minimum expected risk at an insider with exactly k IDS active insiders. Since the expected risk decreases with increase in k, the expected risk measured in the simulations is less than $H(k)$.

In figure 2(c), we plot $H(k)$ as a function of $p + q$ for different values of p and q. We select $k = 3$ here. Note that $H(k)$ is small when $p + q$ is either small or large, and maximum around $p+q = 1$. The plots are symmetric around $p+q = 1$. This is because the aggregation process is most prone to error around $p + q = 1$. When $p + q$ is small (i.e., $p + q << 1$), only few insiders will erroneously determine whether a packet is bad. Thus, if a packet is bad, most insiders will report it as bad, and if a packet is good, most insiders will determine it as good. For $p + q < 1$, the optimum aggregation rule considers a packet as bad if and only if more than a certain number of insiders report it as bad. Thus, the aggregation process is less likely to have errors and hence $H(k)$ is small. Now, when $p+q$ is large (i.e., $p+q >> 1$), a large number of insiders will erroneously determine whether a packet is bad. Thus, if a packet is bad, only few insiders will report it as bad, and if a packet is good, again only few insiders will determine it as good. For $p + q > 1$, the optimum aggregation rule considers a packet as bad if and only if fewer than a certain number of insiders report it as bad. Thus, the aggregation process is again less likely to have errors and hence $H(k)$ is small. When $p+q \approx 1$, for each packet, the number of insiders reporting it as bad and good are similar. Thus, the reports have less correlation with the packet's nature. Hence, the aggregation process is prone to more errors and $H(k)$ is larger.

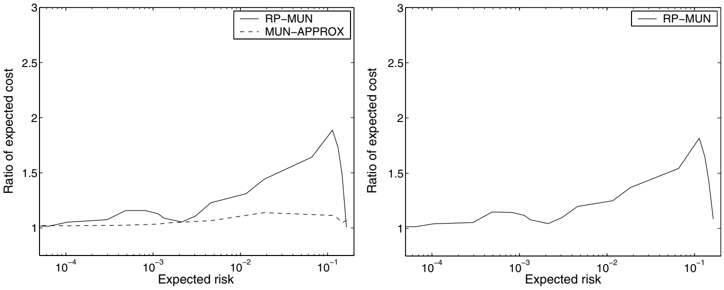
Using simulations, we now compare the efficacy of different algorithms for selecting the IDS active insiders (Figure 3). We consider $p = q = 0.05$. We plot the ratio of the expected detection costs of different algorithms (RP and MUN, and MUN and APPROX) as a function of the expected risk. For each risk value γ and algorithm, we determine the lowest value of k that attains expected risk less than γ, and determine the expected detection costs of the algorithm at this value of k. We consider both static and mobile insider nodes. The intruder and its target are now selected uniformly. Thus the path between them, which is selected by AODV, consists of arbitrary number of hops.

We first consider static insiders (Figure 3(a)). First, note that MUN and APPROX perform similarly all through. Thus, intelligent selection based on a local criteria can perform similar to that based on global criteria in this problem. We now assess MUN's



(a) p+q < 1 (b) p+q > 1 (c) k=3

Fig. 2. In figures (a) and (b) we plot the expected risk as a function of k when $p + q < 1$ and $p + q > 1$ respectively. In figure (c), we plot the expected risk as a function of $p + q$ when k=3



(a) Static insiders (b) Mobile insiders

Fig. 3. We plot the ratio of the expected cost of different algorithms as a function of the expected risk. Here, $p = q = 0.05$. We compare RP and MUN and MUN and APPROX for static insiders in figure (a). We compare RP and MUN for mobile insiders in figure (b)

advantage over the naive RP. For intermediate values of acceptable risk, MUN reduces the detection cost of RP by about half. In this region, the risk values require intermediate values of k, and MUN's intelligent selection of IDS active nodes attains the same coverage as RP while using half the number of IDS active insiders as RP. For very low expected risks, RP and MUN have similar detection costs. This is because k needs to be large in this case, and equals the number of neighbors of the insiders in many cases. But then both RP and MUN would need to execute IDS in a large fraction of insider nodes, e.g., around $70 - 80\%$. Thus both have similar detection costs. Similarly, when the tolerable expected risk is high, both RP and MUN need very few IDS active insiders. Thus, again both have similar detection costs. In both extremes, intelligent coverage algorithms do not provide significant benefits. The trends and conclusions remain similar for mobile insiders (Figure 3(b)). Given that APPROX requires global recomputation every time an insider moves, APPROX can not be used in this case. We have investigated the performance of these strategies in many other scenarios. The results are similar for other transmission radii, mobility parameters, number of insider and intruder nodes, p, q, etc. [11]. We do not show these figures due to lack of space.

Summarizing, MUN's performance is similar to APPROX that has analytical performance guarantees and MUN's performance is significantly better than that of RP. MUN is much simpler than APPROX, but is little more complex than RP. Thus, MUN attains a good balance between complexity and performance for providing robust intrusion detection in ad hoc networks.

5 Conclusion

We consider ad hoc networks with imperfections in the nodes performing intrusion detection tasks. Combining tools from the theories of hypothesis testing and approximation algorithms, we develop a framework to counter different threats while minimizing the resource consumption. We obtain computationally simple optimal rules for aggregating and thereby minimizing the errors in the decisions of the nodes detecting the intrusion. We also show that the optimal selection of nodes that execute the detection modules is an NP-hard problem. Hence, we present a polynomial complexity selection algorithm APPROX that attains a guaranteeable approximation bound. This algorithm is modified to allow for seamless operation in time varying topologies. The modified version (MUN) needs only simple computations and local message exchanges when nodes move. Nevertheless, our simulation reveals that MUN's performance is similar to that of APPROX. In many cases MUN reduces the detection cost by about half as compared to a naive heuristic. We conclude that MUN provides a good balance between complexity and performance for attaining robust intrusion detection in ad hoc networks.

References

1. Cole, E.: Hackers Beware. New Riding Publishing (2001)
2. Cheswic, W., Bellovin, W.: Firewalls and Internet Security. Addison Wesley (1999)
3. Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., Paxson, V.: Stream control transmission protocol. RFC 2960 (2000)
4. Nikita Borisov, Ian Goldberg, D.W.: Intercepting mobile communications: The insecurity of 802.11. In: Proceedings of MOBICOM 2001. (2001)
5. Denning, D.: An intrusion detection model. In: IEEE Transactions on Software Engineering. Volume SE-13. (2001) 222–232
6. Garfinkel, S., Spafford, G.: Practical UNIX and Internet Security. 2nd edn. O'Reilly and Associates (1996)
7. Ko, C., Brutch, P., Rowe, J., Tsafnat, G., Levitt, K.: System health and intrusion monitoring using a hierarchy of constraints. In: 4th International Symposium, Recent Advances in Intrusion Detection. (2001) 190–204
8. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: Mobile Computing and Networking. (2000)
9. Subhadrabandhu, D., Sarkar, S., Anjum, F.: Efficacy of misuse detection in adhoc networks. In: Proceedings of the IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON). (2004)
10. Zhang, Y., Lee, W.: Intrusion detection in wireless ad-hoc networks. In: Proceedings of the Sixth International Conference on Mobile Computing and Networking (MobiCom). (2000)

11. Subhadrabandhu, D., Sarkar, S., Anjum, F.: Misuse detection with imperfect defenders in adhoc networks. Technical report, University of Pennsylvania Technical Report, http://www.seas.upenn.edu/∼swati/publication.htm (2004)
12. Ramanujan, R., Kudige, S., Nguyen, T., Takkella, S., Adelstein, F.: Intrusion-resistant ad hoc wireless networks. In: Proceedings of MILCOM. (2002)
13. Ptacek, T., Newsham, T.: Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, an SNI Technical Report, http://www.aciri.org/vern/Ptacek-Newsham-Evasion-98.ps (1998)
14. Dharmapurikar, S., Krishnamurthy, P., Sproull, T., Lockwood, J.: Deep packet inspection using parallel bloom filters. In: Micro, IEEE. Volume 24. (2004) 52–61
15. McHugh, J.: Intrusion and intrusion detection. International Journal of Information Security (2001)
16. Poor, H.V.: An Introduction to Signal Detection and Estimation. 2nd edn. Springer-Verlag (1994)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability. W. H. Freeman and Company (2000)
18. Vazirani, V.: Approximation Algorithms. Springer (2001)

Self-configurable Key Pre-distribution in Mobile Ad Hoc Networks

Claude Castelluccia[1,⋆], Nitesh Saxena[2], and Jeong Hyun Yi[2,⋆⋆]

[1] INRIA, 655, avenue de l'Europe,
38330 Montbonnot, France
claude.castelluccia@inrialpes.fr
[2] School of Information and Computer Science,
University of California at Irvine,
Irvine, CA 92697, USA
{nitesh, jhyi}@ics.uci.edu

Abstract. We present two new schemes that, in the absence of a centralized support, allow a pair of nodes of a mobile ad hoc network to compute a shared key without communicating. Such a service is important to secure routing protocols [1, 2, 3]. The schemes are built using the well-known technique of threshold secret sharing and are secure against a collusion of up to a certain number of nodes.

We evaluate and compare the performance of both the schemes in terms of the node admission and pairwise key establishment costs.

1 Introduction

Mobile ad hoc networks (MANETs) are, by their very nature, vulnerable to many types of attacks. The security of MANETs is often predicated on the availability of efficient key management techniques. However, the usual features of: (1) lack of a centralized authority and (2) dynamic nature of MANETs, represent major obstacles to providing secure, effective and efficient key management. What further complicates the issue is that, in many applications (such as secure routing [1, 2, 3]) cryptographic keys need to be established *prior* to communication. As a result, standard key exchange solutions, e.g., Station-to-Station protocol [4], are not appropriate since: (1) they require the nodes to interact and (2) they rely on some form of a Public Key Infrastructure (PKI) which is not usually available in MANETs. Related to the latter is the underlying use of public key cryptography which is too expensive for some mobile devices.

Contributions: This paper proposes two efficient, fully distributed and secure key management solutions for MANETs. The solutions, collectively referred to as *Threshold Key Pre-distribution*, allow nodes in a MANET to establish pairwise

[⋆] This work has been done while visiting UC Irvine.

[⋆⋆] corresponding author.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1083–1095, 2005.

keys without communicating and without the need of a PKI. The first scheme, called Matrix Threshold Key Pre-distribution (*MTKP*), results from the blending of two well-known techniques: Blom's key pre-distribution [5, 6] and threshold secret sharing [7], and the second scheme, referred to as Polynomial Threshold Key Pre-distribution (*PTKP*), employs threshold secret sharing using a polynomial. In both *MTKP* and *PTKP*, a node joins a MANET by receiving a secret token from t different nodes, where t is a security parameter. The schemes are *auto-configurable* in the sense that there is no centralized support required and a node becomes a member only if it is approved by at least t member nodes. Once a node becomes member, it can compute a secret key with any other member without interaction. The proposed schemes are secure against collusion of up to a certain number $(t-1)$ of compromised nodes.

The contribution of this paper is not limited to just the design of efficient key distribution schemes. We also demonstrate our claims of efficiency via extensive analysis and experiments. The schemes have been implemented and tested in a real MANET setting and their performance is compared and analyzed in detail.

Organization: The rest of this paper is organized as follows: Section 2 overviews the related work. Section 3 provides some background on necessary cryptographic building blocks. Sections 4 and 5 present our Threshold Key Pre-distribution schemes *MTKP* and *PTKP* respectively. We discuss some security and other relevant issues of the proposed schemes in Section 6. Finally, in Section 7, we describe the implementation and the performance of our schemes.

2 Related Work

Key distribution can be easily achieved if we assume the existence of a PKI. However, this assumption is not realistic in many MANET environments. Zhou and Haas [8] proposed to distribute a Certification Authority (CA) service among several nodes of the network. Although attractive, this idea is not applicable to MANETs. Their approach is hierarchical: only selected nodes can serve as part of the certification authority and thus take part in admission decisions. Moreover, contacting the distributed CA nodes in a MANET setting is difficult since such nodes might be many hops away.

In a related result, Kong, et al. [9] developed an interesting Threshold-RSA (TS-RSA) scheme specifically geared for MANETs. Unfortunately, as pointed out in [10, 11], TS-RSA is neither verifiable nor secure. An alternative Threshold-DSA (TS-DSA) scheme [10] provides verifiability and, hence, tolerates malicious insiders. However, TS-DSA requires $2t-1$ signers to issue certificates, is heavily interactive and thus become quite inefficient in MANET settings. Moreover, all these solutions require a pair of nodes to perform key exchange protocol to establish shared keys.

Recently, Zhu, et al. [12] proposed a pair-wise key distribution scheme based on the combination of probabilistic key sharing and threshold secret sharing. However, it is assumed that the nodes are *pre-configured* with some secrets before deployment which is not realistic in a typical MANET environment. Furthermore, two nodes need to communicate over several distinct paths to establish a shared key. In contrast, we do not assume any such pre-configuration and do not require nodes to communicate when establishing a secret key.

3 Building Blocks

3.1 Threshold Secret Sharing

(t, n) threshold cryptography allows n parties to share the ability to perform a cryptographic operation in a way that any t parties can perform this operation jointly, whereas no coalition of up to $t-1$ parties can do so. We use Shamir's secret sharing scheme [7] which is based on polynomial interpolation. To distribute shares among n users, a trusted dealer TD chooses a large prime q, and selects a polynomial $f(x) = S + a_1 x + \cdots + a_{t-1}x^{t-1}$ over $\mathbb{Z}_q$ of degree $t-1$ such that $f(0) = S$, where S is the group secret. The TD computes each user's share ss_i such that $ss_i = f(id_i) \pmod q$, and securely transfers ss_i to user M_i. Then, any group of t members who have their shares can recover the secret using the Lagrange interpolation formula: $f(0) = \sum_{i=1}^{t} ss_i \; l_i(0) \pmod q$, where $l_i(0) = \prod_{j=1, j\neq i}^{t} \frac{id_j}{id_j - id_i} \pmod q$. To enable the verification of the secret shares, TD publishes a commitment to the polynomial as in *Verifiable Secret Sharing* (VSS) [13]. VSS setup involves a large prime p such that q divides $p-1$ and a generator g which is an element of $\mathbb{Z}_p^*$ of order q. TD computes w_i $(i = 0, \cdots, t-1)$, called the **witness**, such that $w_i = g^{a_i} \pmod p$ and publishes these w_i-s in some public domain (e.g., a directory server). On receiving the secret share ss_i from M_i, M_j verifies the correctness of ss_i by checking $g^{ss_i} = \prod_{k=0}^{t-1} (w_k)^{{id_i}^k} \pmod p$.

3.2 Blom's Key Pre-distribution

Blom proposed a key pre-distribution scheme that allows any pair of users in a group to compute a pairwise key without communicating [5]. This scheme is secure unless λ users collude (the parameter λ will be defined later). If less than λ users collude, then it is proven that the system is completely secure i.e., the colluding nodes can not compute any pair-wise keys other than their own. However, if λ or more users collude, the whole group is compromised and the colluding users can compute the pair-wise keys of all other members.

In Blom's proposal, a trusted dealer TD computes a $\lambda \times N$ matrix B over $\mathbb{Z}_q$, where N is the maximum size of the group, q is a prime, and $q > N$.

One example of such a matrix is a Vandermonde matrix whose element $b_{ij} = (g^j)^i \pmod q$ as seen below, where g is the primitive element of $\mathbb{Z}_q^*$.

$$B = \begin{bmatrix} b_{ij} = (g^j)^i \\ \text{for } i, j = 1, \cdots, \lambda \end{bmatrix} \pmod q$$

Note that this construction requires that $N\lambda < \phi(q)$ i.e., $N\lambda < q - 1$.

Since B is a Vandermonde matrix, it can be shown that any λ columns are linearly independent when $g, g^2, g^3, ..., g^N$ are all distinct [14]. The TD then creates a random $\lambda \times \lambda$ symmetric matrix D over $\mathbb{Z}_q$, and computes an $N \times \lambda$ matrix $A = (DB)^T$, where T indicates a transposition of the matrix.

The matrix B is published while the matrix D is kept secret by the TD. Since D is symmetric, the *key matrix* $K = AB$ is also symmetric

$$K = (DB)^T B = B^T D^T B = B^T DB = (AB)^T = K^T.$$

This shows that K is also a symmetric matrix.

Every user in the group is given a row (corresponding to its identifier) of the matrix A. The user M_i multiplies its row of the matrix A with the j^{th} column of the matrix B to establish a shared key with another user M_j.

4 MTKP: Matrix Threshold Key Pre-distribution

The different steps of the *MTKP* scheme, which is based on Blom's key predistribution described in previous section, are summarized as follows. (A more precise description of each of these steps is presented in the following subsections.)

1. *Bootstrapping:* The network is bootstrapped by either one single founding member or a set of founding members. The founding members compute the matrices D, B and A, defined in Section 3.2. The founding members then split the matrix D into n shares, $ss_i(D)$ for $i = 1, \cdots, n$, such that at least t shares are required to reconstruct it. Each node receives a share of D.
2. *Member Admission:* A prospective member M_η initiates the admission protocol by sending a JOIN_REQ message to the network. A member node, that receives this JOIN_REQ message and approves the admission of M_η, replies, over a secure channel (refer to Section 6), with the row η of its share of matrix A (we explain in the following section how a share of the matrix A can be computed from a share of the matrix D). Once M_η receives shares from at least t different nodes, it can retrieve the row η of matrix A using Lagrange interpolation. It can then use these system secrets to compute a key with any other member of the network according the Blom key establishment protocol described in Section 3.2. Each joining node is also provided with *partial* shares of the whole matrix D from t current member nodes, which it can use to reconstruct its *share* of the matrix A, and consequently use it to admit new members.
3. *Secret Key Computation:* The pair-wise key computation procedure is the same as the one described in Section 3.2.

Note that our scheme is completely distributed and as such can be qualified as a peer-to-peer scheme; nodes get admitted by a quorum of their peers. The network can get bootstrapped by a set of nodes that get together and compute the security parameters of the network in a distributed way. Our scheme does not require any kind of central authority or trusted third party.

4.1 Bootstrapping

In *MTKP* scheme, a network can be bootstrapped (i.e., initialized) by one node (centralized bootstrapping) or a set of t or more nodes (distributed bootstrapping).

Centralized Bootstrapping. The centralized bootstrapping proceeds as follows. First, a founding member FM generates the network parameters, namely N, λ, t, q, p, g, and the matrices $D = [d_{ij}]$ and $B = [b_{ij}]$, where N is the maximum numbers of nodes in the network, (λ, q, p, g) are the security parameters, B is $\lambda \times N$ public matrix such that $b_{ij} = (g^j)^i \pmod{q}$ for $i, j \in [1, \lambda]$, and D is $\lambda \times \lambda$ symmetric matrix of secrets.

Next, the FM publishes $(N, \lambda, t, q, p, g, B)$ in some public directory, but keeps D secret. It then computes the matrix A such that $A = (DB)^T$ and sends a share of the whole matrix A to each node.

To compute the share $ss_v(D)$ for member M_v, FM selects polynomials for each element d_{ij} of $\lambda \times \lambda$ matrix D. Each polynomial is defined as follows; $f_{d_{ij}}(x) = \sum_{\alpha=0}^{t-1} \delta_{ij}^{(\alpha)} \cdot x^{\alpha} \pmod{q}$ such that $\delta_{ij}^{(0)} = d_{ij}$. The share of matrix D is made up of shares of its elements $ss_v(d_{ij})$. In other words, $ss_v(D) = [ss_v(d_{ij})] = [f_{d_{ij}}(v)]$ for $i, j = 1, \cdots, \lambda$.

As for $r_v(A)$ such that $r_v(A) = [a_{vj}]$ for $j = 1, \cdots, \lambda$, each element of $r_v(A)$ is simply computed by FM since it knows the secret matrix D. That is, $a_{vj} = \sum_{\beta=1}^{\lambda} d_{j\beta} \cdot b_{\beta v} \pmod{q}$. Then FM distributes $ss_v(D)$ and $r_v(A)$ to each AN_v.

Distributed Bootstrapping. The network can alternatively be bootstrapped by a set of t or more founding members. The secret matrix D can be generated in fully distributed manner. Note that in the centralized mode, the FM is similar to a trusted third party and is, therefore, a single point of failure. In this proposal, a group of members (the founding members in our scenario) collectively compute shares corresponding to Shamir secret sharing of a random value without a centralized trusted dealer. This procedure is so-called *Joint Secret Sharing (JSS).* For more details, refer to [15].

4.2 Member Admission

In order to join the network, a prospective node M_η must collect at least t shares of matrix A's row η from the current member nodes and a valid share of the whole matrix D. Figure 1 shows the protocol message flow for the member admission process.[1]

[1] In order to secure the protocol against common *replay* attacks [4], we note that it is necessary to include timestamps, nonces and protocol message identifiers. However, in order to keep our description simple, we omit these values.

$$msg1(M_\eta \rightarrow M_\nu)\text{: } REQ = \{id_\eta, y_\eta\},\ S_\eta(REQ) \tag{1}$$
$$msg2(M_\eta \leftarrow M_\nu)\text{: } REP = \{id_\nu, y_\nu\},\ S_\nu(REP, H(REQ)) \tag{2}$$
$$msg3(M_\eta \rightarrow M_\mu)\text{: } SL_\eta, MAC(DHK_{\eta\mu},\ H(SL_\eta, msg1, msg2)) \tag{3}$$
$$msg4(M_\eta \leftarrow M_\mu)\text{: } E_{DHK_{\eta\mu}}\{pss_\mu(r_\eta(D)), ss_\mu(r_\eta(A))\} \tag{4}$$

Fig. 1. *MTKP* Admission Protocol

1. M_η sends to at least t current member nodes M_ν-s ($\nu \in \{1, n\}$) a signed `JOIN_REQ` message which contains his identity id_η and his public Diffie-Hellman (DH) component $y_\eta (= g^{x_\eta} \mod p)$. The details about how id_η is generated and verified are discussed in Section 6.
2. After verifying the signed `JOIN_REQ`, the member nodes who wish to participate in the admission process of M_η reply with a signed message containing their respective values id_ν and y_ν.
3. M_η selects t sponsors $M_\mu (\mu \in_R \nu, |\mu| = t)$, computes a secret key $DHK_{\eta\mu}$ with each of them, forms a sponsor list SL_η which contains the ID-s of the t selected sponsors, and replies with an authenticated acknowledgment message to each of them.
4. Each sponsoring node (M_μ) on receiving $msg3$, computes the secret key $DHK_{\eta\mu}$ and replies with row η of it share of the matrix A, $ss_\mu(r_\eta(A))$. The elements of $ss_\mu(r_\eta(A))$ are computed as $ss_\mu(r_\eta(a_{\eta j})) = \sum_{\beta=1}^{\lambda} ss_\mu(d_{j\beta}) \cdot b_{\beta\eta}$ (mod q), for $j = 1, \cdots, \lambda$. This message is encrypted with $DHK_{\eta\mu}$. Each (M_μ) also responds with the *shuffled* partial share of matrix D, $pss_\mu^\eta(D)$, such that $pss_\mu^\eta(D) = [pss_\mu^\eta(d_{ij})] = [ss_\mu(d_{ij}) \cdot l_\mu(\eta)]$ (mod q) for $i, j = 1, \cdots, \lambda$. This message is also encrypted using $DHK_{\eta\mu}$.

   Note that the Lagrange coefficients $l_\mu(\eta)$ are publicly known, and therefore, M_η can derive $ss_\mu(d_{ij})$ from $pss_\mu^\eta(d_{ij})$. This can be prevented using the *shuffling* technique proposed in [9] by adding extra random value R_{ij} to each share. These R_{ij}-s are secret values and must sum up to zero by construction. They must be securely shared among the t sponsoring nodes.
5. M_η decrypts the messages it receives from the different nodes and calculates his own $r_\eta(A)$ by adding up all $ss_\mu(r_\eta(A))$-s as follows: $r_\eta(A) = \sum_{\mu=1}^{t} ss_\mu(r_\eta(A)) \cdot l_\mu(0) = [\sum_{\mu=1}^{t} ss_\mu(r_\eta(a_{\eta j})) \cdot l_\mu(0)]$ (mod q) for $j = 1, \cdots, \lambda$. M_η also calculates his own share of the matrix D, $ss_\eta(D)$, by adding up the partial share values such that $ss_\eta(D) = \sum_{\mu=1}^{t} pss_\mu^\eta(D) = [\sum_{\mu=1}^{t} pss_\mu^\eta(d_{ij})]$ (mod q) for $i, j = 1, \cdots, \lambda$.

4.3 Secret Key Computation

When a node, M_i, reconstructs its private row of matrix A, $r_i(A) = [a_{i1}, \cdots, a_{i\lambda}]$, he can compute a secret key, K_{ij}, with any other node, M_j, of the network as follows:

Since $a_{ij} = \sum_{\alpha=1}^{\lambda} d_{j\alpha} \cdot b_{\alpha i}$ and $d_{ij} = d_{ji}$,

$$K_{ij} = \sum_{\beta=1}^{\lambda} a_{i\beta} b_{\beta j} = \sum_{\beta=1}^{\lambda} \sum_{\alpha=1}^{\lambda} d_{\beta\alpha} b_{\alpha i} b_{\beta j} = \sum_{\alpha=1}^{\lambda} \sum_{\beta=1}^{\lambda} d_{\alpha\beta} b_{\beta j} b_{\alpha i} = \sum_{\alpha=1}^{\lambda} a_{j\alpha} b_{\alpha i} = K_{ji}.$$

Note that these keys do not have to be computed in advance but can be computed *on-the-fly*.

5 PTKP: Polynomial Threshold Key Pre-distribution

The various steps of the *PTKP* scheme, which is based on polynomial secret sharing, are summarized as follows.

1. *Bootstrapping:* The network is bootstrapped by either one single founding member or a set of founding members. The founding member(s) compute the secret share polynomial $f(z)$, defined in Section 3.1. Every member is provided with a share of this polynomial.
2. *Member Admission:* A prospective member M_η initiates the protocol by sending a `JOIN_REQ` message to the network. A member node, that receives this `JOIN_REQ` message and approves the admission of M_η, replies, over a secure channel (refer to Section 6), with *partial* shares of the polynomial which it can use to reconstruct its *share* of polynomial, and consequently use it to admit new members and establish pairwise keys with other members.
3. *Secret Key Computation:* Each node uses its secret share and the public VSS information (as described in Section 3.1) to compute pairwise keys with other nodes.

5.1 Bootstrapping

Centralized Bootstrapping. The centralized bootstrapping works exactly as described in Section 3.1.

Distributed Bootstrapping. A group of t or more founding members employ JSS [15] to collectively compute shares corresponding to Shamir secret sharing of a random value.

5.2 Member Admission

In order to join the network, a prospective node M_η must collect at least t partial shares from existing nodes to be able to compute its secret share. Figure 2 shows the protocol message flow for the member admission process.

1-3. Steps 1-3 are exactly the same as in the *MTKP* admission protocol described in Section 4.2.

4. Each sponsoring node (M_μ) on receiving $msg3$, computes the secret key $DHK_{\eta\mu}$ and replies with the *shuffled* partial share[9], $pss_\mu(\eta)$, such that $pss_\mu(\eta) = ss_\mu \cdot l_\mu(\eta) \pmod q$. This message is encrypted using $DHK_{\eta\mu}$.

$$\begin{array}{|ll|}\hline msg1(M_\eta \rightarrow M_\nu)\text{: } REQ = \{id_\eta, y_\eta\},\ S_\eta(REQ) & (1) \\ msg2(M_\eta \leftarrow M_\nu)\text{: } REP = \{id_\nu, y_\nu\},\ S_\nu(REP, H(REQ)) & (2) \\ msg3(M_\eta \rightarrow M_\mu)\text{: } SL_\eta, MAC(DHK_{\eta\mu},\ H(SL_\eta, msg1, msg2)) & (3) \\ msg4(M_\eta \leftarrow M_\mu)\text{: } E_{DHK_{\eta\mu}}\{pss_\mu(\eta)\} & (4) \\ \hline \end{array}$$

Fig. 2. *PTKP* Admission Protocol

5. M_η decrypts the messages it receives from the different nodes and calculates his own secret share ss_η, by adding up the partial share values such that $ss_\eta = \sum_{\mu=1}^{t} pss_\mu(\eta)$.

5.3 Secret Key Computation

Any pair of nodes M_i and M_j can establish shared keys using their respective secret shares ss_i, ss_j and the public VSS information as described in Section 3.1. M_i computes $g^{ss_j} = \prod_{k=0}^{t-1}(w_k)^{{id_j}^k} \pmod p$ from the public commitment values, and exponentiate it to its own share ss_i to get a key $k_{ij} = (g^{ss_j})^{ss_i} \pmod p$. Similarly, M_j computes $g^{ss_i} = \prod_{k=0}^{t-1}(w_k)^{{id_i}^k} \pmod p$ and exponentiate it to its own share ss_j to get a key $k_{ji} = (g^{ss_i})^{ss_j} \pmod p$. Since, $k_{ij} = k_{ji}$, M_i and M_j have a shared secret key.

The above scheme remains secure under the *Computational Diffie-Hellman* (CDH)[2] assumption. In other words, an adversary who corrupts at most $t-1$ nodes, can not compute a shared key between any pair of uncorrupted nodes, as long as the CDH assumption holds.

6 Discussion

Identifier Configuration. In the *MTKP* and *PTKP* schemes, the identifier id_i of each node M_i must be *unique* and *verifiable*. Otherwise, a malicious node could use the identifier of some other node and get its secret from the member nodes during the admission process.

For unique and unforgeable ID assignment, we propose to use a solution based on *Crypto-Based ID (CBID)* [16]: The id_i is chosen by the node itself from an ephemeral public/private key pair. More specifically, the node computes id_i as follows: $id_i = H_{64}(PK_i|NID)$, where PK_i is M_i's temporary public key or DH public key y_i in our schemes, NID is the network identifier and $H_{64}(\cdot)$ a 64-bit long hash function. When a node contacts the member nodes for admission, it sends its identifier id_i together with its ephemeral public key PK_i and signs a challenge sent by the member node. Upon reception of the signature, the member node can verify that the id_i actually belongs to the requesting node (by verifying the signature and that the id_i was generated as $H_{64}(PK_i|NID)$. Note

[2] CDH assumption: In a cyclic group generated by $g \in Z_p^*$ of order q, for $a, b \in \mathbb{Z}_q^*$, given $(g, g^a \pmod p), g^b \pmod p)$, it is hard to compute $g^{ab} \pmod p$.

that the PK_i does not need to be certified and therefore no PKI is required. The identifier is *verifiable* because a node that does not know the private key, associated with the public key used to generate an ID, can not claim to own it. Furthermore since i is computed from a hash function, collision probability between two nodes is very low. As a result, the identifier are *statistically unique.* Note that this solution requires that $N = 2^{64}$. However, as we will see this has no effect on the performance or scalability of our proposal.

Secure Channel Establishment. In the proposed admission protocols, the channels between the node requesting admission and each of the member nodes must be authenticated and encrypted. It has to be authenticated because each member node must be sure that it is sending the shares to the correct node (i.e., the node that claims to own the identifier). Otherwise, the member node could send the shares to an impersonating node. Similarly, the joining node also needs to authenticate the member nodes. The channel has to be private because otherwise a malicious node that eavesdrops on the shares sent to a node could reconstruct the node's secret and impersonate it.

Establishing an authenticated and private channel usually requires the use of certificates, which bind identities to public keys, and an access to a PKI. However, PKI is not always available in MANET environments. Fortunately in our case, what is really needed is a way to bind an identifier to a public key, where the identifier is a number that identifies one row of the matrix A. This binding is actually provided by *CBID*, described previously. As a result, certificates and PKI are not required. Therefore, the PKs that are sent in message 1 and 2 of the protocols described in Sections 4.2 and 5.2 do not need to be certified.

Parameters Selection. The security of the *MTKP* scheme relies on two security parameters t and λ, whereas the *PTKP* scheme depends only on t. λ and t denote the number of collusions needed to break these schemes. These parameters should be selected carefully. In particular, it is suggested to set $\lambda = t$. However, more generally, λ should be at least t in the hierarchical MANET settings where only a subset of nodes possesses the ability to admit new nodes. For the evaluation of our schemes (as described in the next section) we set $\lambda \geq t$.

DoS Resistance. A malicious node can easily launch a DoS (Denial-of-Service) attack toward a candidate node by inserting incorrect secret shares. This attack would actually deny or disrupt the service to legitimate nodes. To deal with this important problem a node must be able to verify the validity of its reconstructed secrets (i.e., its row of the matrix A and its share of the whole matrix D in the *MTKP* scheme and its secret share in the *PTKP* scheme) before using them. This can be done with *Verifiable Secret Sharing (VSS)* [13], as detailed in an extended version of this paper [17].

7 Performance Evaluation

We implemented both *MTKP* and *PTKP* protocols and evaluated them in a real MANET environment in terms of node admission and pairwise key computation costs.

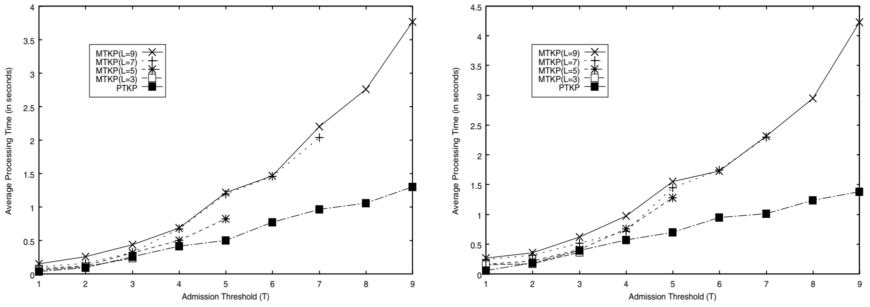
7.1 Experimental Setup

The *MTKP* and *PTKP* protocol suite is implemented on top of the OpenSSL library [18]. It is written in C for Linux, and consists of about 10,000 lines of code for each. The source code is available at [19].

For the experimental set-up, we used a total of five laptops; four laptops with a Pentium-3 800MHz CPU and 256MB memory and one laptop with a Mobile Pentium 1.8 GHz CPU and 512MB memory. Each device ran Linux 2.4 and was equipped with a 802.11*b* wireless card configured in ad-hoc mode. Specifically, for measuring the admission cost, four laptops with same computing power were used to configure the existing member nodes and the high-end laptop was used for the joining node. In our experiments, each node (except the joining node) was emulated by a daemon and each machine was running up to three daemons. The measurements were performed with the different threshold values t and λ for *MTKP*. The size of the parameters q was set to 160-bits and p to 512-bits or 1024-bits.

7.2 Admission Cost

To evaluate the admission cost, we measured the total processing time between the sending of the `JOIN_REQ` by the prospective node and the receiving (plus verification) of acquired credentials (i.e., $r_\eta(A)$ and $ss_\eta(D)$ in *MTKP* and ss_η in *PTKP*). The resulting measurements include the average computation time of the basic operations, the communication costs such as packet encoding and decoding time, the network delay, and so on.



(a) $|p|$=512

(b) $|p|$=1024

Fig. 3. Admission Cost

Figure 3 shows the average admission time for the joining node for different values of the threshold t. For the *MTKP* testing, λ was set to 3, 5, 7 and 9. (In the figure, λ is denoted by L.)

As observed from the graphs, the cost for a node to join the network with *PTKP* is cheaper than that of *MTKP*. This difference in the costs between *MTKP* and *PTKP* is even higher for higher threshold values. The reason is quite intuitive: *MTKP* requires more computation and bandwidth than *PTKP*. More specifically, the *MTKP* scheme requires $O(\lambda^2 t)$ *multiplications* and $O(\lambda^2)$ *exponentiations* whereas *PTKP* requires only $O(t)$ *multiplications* and $O(1)$ *exponentiations*. For the bandwidth costs, refer to Table 1.

Table 1. Bandwidth Comparison

	MTKP	PTKP
Admission	$O(\lambda t\|q\|) + O(\lambda^2\|q\|)$	$O(t\|q\|)$
Shuffling	$O(\lambda^2 t^2\|q\|)$	$O(t^2\|q\|)$

This table shows that the *PTKP* scheme is very efficient in terms of bandwidth. This is an important property for MANET systems which consist of battery-operated devices, because wireless transmission is considered as the most energy consuming operation.[3]

7.3 Key Computation Cost

Table 2 compares the cost of computing a pair-wise key in our schemes. The results show that *MTKP* performs significantly better than a *PTKP* protocol. The achieved gains with $\lambda = 9$ range from 10 ($t = 1$) to 13 ($t = 9$), and from 305 to 307 for 512-bit and 1024-bit p, respectively. In other words, *MTKP* is 10 to 307 times faster than *PTKP* when establishing a shared secret key.

These results were actually expected because in *MTKP* the pair-wise computation requires only $O(\lambda)$ modular *multiplications* where the modulus size is 160 bits. In contrast, *PTKP* requires $O(t)$ expensive modular *exponentiations* with a modulus size of 512 or 1024 bits.

8 Conclusion

We presented *Threshold Key Pre-distribution*: distributed solutions to the key pre-distribution problem in MANETs. Our solutions, *MTKP* and *PTKP*, are based on the secret sharing techniques and are secure against collusive attacks by a certain threshold of nodes. The solutions allow any pair of nodes in the network to establish shared keys *without communication*, as opposed to the standard

[3] It has been shown that sending one bit of data is roughly equivalent to adding 1000 32-bit numbers [19].

Table 2. Key Computation Cost (in msecs, P4-3.0GHz, 1GB Memory)

t	MTKP ($\lambda \geq t$)				PTKP	
	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$	$\lambda = 9$	$\|p\| = 512$	$\|p\| = 1024$
1	0.0371	0.0301	0.0430	0.0550	0.574	17.780
2	0.0398	0.0415	0.0506	0.0570	0.683	18.150
3	0.0436	0.0424	0.0568	0.0564	0.713	18.180
4	-	0.0365	0.0595	0.0655	0.663	18.220
5	-	0.0431	0.0565	0.0629	0.753	18.370
6	-	-	0.0628	0.0563	0.772	18.450
7	-	-	0.0562	0.0629	0.782	18.570
8	-	-	-	0.0644	0.851	18.540
9	-	-	-	0.0637	0.871	19.120

Diffie-Hellman key exchange protocols. We implemented the *MTKP* and *PTKP* schemes and evaluated them in real MANET setting. Our analysis show that *MTKP* fares better than *PTKP* as far as the pairwise key establishment costs are concerned. However, in terms of the node admission costs, the latter outperforms the former. Based on this analysis, we conclude that the *MTKP* scheme is well-suited for MANET applications where node admission is not a frequent operation, whereas the *PTKP* scheme is more applicable for highly dynamic MANETs consisting of mobile devices with reasonably high computation power.

References

1. Hu, Y.C., Perrig, A., Johnson, D.B.: Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In: ACM MobiCom. (2002) 12–23
2. Hu, Y.C., Johnson, D.B., Perrig, A.: SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In: IEEE WMCSA. (2002) 3–13
3. Papadimitratos, P., Haas, Z.: Secure Routing for Mobile Ad Hoc Networks. In: SCS CNDS. (2002)
4. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press. (1997) ISBN 0-8493-8523-7.
5. Blom, R.: An Optimal Class of Symmetric Key Generation Systems. In: EUROCRYPT '84. LNCS, IACR (1984)
6. Leighton, T., Micali, S.: Secret-Key Agreement without Public-Key Cryptography. In: CRYPTO'93. (1993)
7. Shamir, A.: How to Share a Secret. Communications of the ACM **22** (1979) 612–613
8. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network Magazine **13** (1999) 24–30
9. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing Robust and Ubiquitous Security Support for MANET. In: IEEE ICNP. (2001)
10. Narasimha, M., Tsudik, G., Yi, J.H.: On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In: IEEE ICNP. (2003) 336–345

11. Jarecki, S., Saxena, N., Yi, J.H.: An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In: ACM SASN. (2004) 1–9
12. Zhu, S., Xu, S., Setia, S., Jajodia, S.: Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach. In: IEEE ICNP. (2003)
13. P.Feldman: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: FOCS. (1987) 427–437
14. MacWilliams, F.J., Sloane, N.: The Theory of Error-Correcting Codes. North Holland, Amsterdam (1997)
15. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In Davies, D., ed.: EUROCRYPT '91. Number 547 in LNCS, IACR (1991) 552–526
16. Montenegro, G., Castelluccia, C.: Crypto-based Identifiers (CBIDs): Concepts and Applications. ACM TISSEC **7** (2004)
17. Castelluccia, C., Yi, J.H.: DoS-Resistant Self-Keying Mobile Ad-Hoc Networks. Technical report, INRIA-5373, http://sconce.ics.uci.edu/gac/publication.html (2004)
18. OpenSSL Project: (`http://www.openssl.org/`)
19. Peer Group Admission Control Project: (`http://sconce.ics.uci.edu/gac`)

Efficient Use of Route Requests for Loop-Free On-demand Routing in Ad Hoc Networks ⋆

Hari Rangarajan[1] and J.J. Garcia-Luna-Aceves[1,2]

[1] University Of California at Santa Cruz,
Computer Engineering Dept., Santa Cruz, CA 95064
[2] Palo Alto Research Center,
3333 Coyote Hill Road, Palo Alto, CA 94304
{hari, jj}@cse.ucsc.edu

Abstract. We present a new loop-free on-demand routing protocol for ad-hoc networks, the *Labeled Successor Routing (LSR)* protocol, which identifies loop-free successors to a destination using route-request labels (RRL). Each route request (RREQ) used during the on-demand destination search process is identified uniquely by a sequence number associated with the issuing source address. Route replies (RREP), which traverse loop-free paths created by RREQs, carry the associated RRL that is stored by nodes along the created successor path to the destination. Without requiring an additional mechanism for loop-freedom (e.g., per destination-sequence numbers or source-routing) LSR allows neighbors of a source to reply to RREQs, avoiding the destination being the only node capable of replying. Simulations results for scenarios consisting of networks of 50 and 100 mobile nodes show that LSR performs comparably or better than the Dynamic Source Routing (DSR) protocol, AODV, and the Optimized Link State Routing (OLSR) protocol.

1 Introduction

Several on-demand routing protocols have been proposed to date for mobile ad hoc networks (MANET). Such protocols establish routes to only those destinations for which there is traffic, and attempt to ensure loop freedom at every instant to limit control overhead. Because of node mobility and the need to setup routes when required, all on-demand routing protocols are characterized by the use of a route request flood to search for the destination. Each route request(RREQ) attempt by a source for a destination is represented by an unique (source address, flooding identifier) pair, to which we refer as a Route Request Label (RRL). RRLs are required to prevent RREQs from being processed multiple times, and to set up loop-free paths along which route replies generated by intermediate nodes or destinations traverse. However, the actual establishment and maintenance of routes in current on-demand routing protocols is independent of RRLs.

⋆ This work was funded in part by the Baskin Chair of Computer Engineering at UCSC and by the National Science Foundation under Grant CNS-0435522.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1096–1107, 2005.

Current on-demand protocols use RRLs during route request floods, but use additional mechanisms to achieve loop-free routing. For example, the dynamic source routing protocol (DSR) [4] collects partial topology information during the RREQ flood, and uses path information collected to source-route data packets. The feasible label routing (FLR) [5] protocol collects path information much like DSR does, but achieves loop-free hop-by-hop routing of data packets based on just destination addresses by assigning lexicographically ordered labels, derived from path information, to nodes along a successor path to the destination. The Ad hoc On-demand Distance Vector protocol (AODV) [6] uses per-destination sequence numbers to maintain loop-freedom.

Recent work in the MANET Working Group of the Internet Engineering Task Force has focused on generic routing frameworks that are simple to implement, deploy, and test, while additional features can be added as necessary for improved performance. The most recent proposal as of this writing is the Dynamic MANET On-demand [7] Routing Protocol. It is based on AODV and allows additional capabilities to be added by allowing flexible control message parsing; however, DYMO attempts to maintain loop-free routing using the same per-destination-sequence numbers used in AODV. As a result, it is susceptible to the same performance and robustness issues found in AODV, namely: (a) most route requests have to be answered by the destination, and (b) it can suffer from temporary loops, de-facto partitions and count-to-infinity [2, 3].

In this paper we present a new loop-free routing approach that uses the information already needed in route requests to establish and maintain loop-free routes, and allows nodes other than the destinations to initiate route replies. Section 2 presents the design of the Labeled Successor Routing (LSR) protocol, which demonstrates the use of RRLs to achieve efficient loop-free routing. The basic idea behind LSR is very simple: A source floods a route request identified by an unique RRL, which creates a tree rooted at the source as the RREQs are processed only once by each node. When the RREPs traverse loop-free reverse paths, nodes update their routing tables for the specified destination and store the RRL. If the source stored a RRL that belonged to a different source before issuing this new RREQ, it sequences to a new RRL; otherwise, it retains the previously stored RRL. At a later time, the source can pick any neighbor node that has a stored RRL that is more recent or the same as the one stored at the source, as a loop-free successor towards the destination. Section 3 illustrates the working of LSR with an example. Section 4 analyzes the correctness of LSR. Section 5 compares the performance of LSR against AODV, DSR and a proactive link state protocol (the Optimized Link State Routing or OLSR) [1]. Section 6 provides our concluding remarks.

2 Labeled Successor Routing (LSR)

The labeled successor routing (LSR), is an on-demand protocol based on a control signaling (RREQ, RREP and RERR) similar to that used in other on-demand routing protocols.

Each route request label RRL is a unique pair $\{src, id\}$, where src is a node address identifier and id is an integer created by src. The notation src^{RRL} is used to refer to the value of src in a RRL. An empty RRL is denoted by ϕ.

Route replies, and route requests are denoted by super-scripts req, and rep, respectively. For a destination D, node A stores a route-request label, RRL_D^A. The successor to destination D at A is denoted by s_D^A. $NRRL_D^{req}$ is the neighbor-query RRL of the request used to identify neighbors of the source that have a loop-free path to the destination. d_D^A is the distance (hop count) to the destination. lc_A^B is the link cost from node A to B. $DInit_D^{rep}$ is carried in a RREP to indicate whether the route reply was initiated by the destination.

The following relational operator is defined on two labels, RRL_1= $\{src_1$,$id_1\}$, and RRL_2= $\{src_2$,$id_2\}$, to establish ordering among RRLs:

$$RRL_1 \prec RRL_2 \;\; if \;\; src_1 = src_2 \;\wedge id_1 \geq id_2 \tag{1}$$

The above relational operator allow a source to compare if a neighbor can be chosen as a loop-free successor safely (i.e., without incurring a loop). LSR uses the following four conditions based on this operator to maintain loop-free paths:

ASC: (Accept Successor Condition). When node A receives a RREP from node B for destination D, then node A sets $s_D^A \leftarrow B$, if (i) $DInit_D^{rep} = 0$, $src^{RRL_D^A}$= A, and $RRL_D^{rep} \prec RRL_D^A$; or (ii) $DInit_D^{rep} = 1$. If $RRL_D^A = \phi$ or $src^{RRL_D^A} \neq A$, then node A should accept a RREP only if $DInit_D^{rep} = 1$ (i.e., generated by the destination).

SSC: (Start Successor Condition). Node I can issue a RREP responding to a RREQ for destination D if I has an active route to D, and $RRL_D^I \prec NRRL_D^{req}$. If node $I = D$, then it must set $DInit_D^{rep} = 1$ and issue a RREP.

RSC: (Relay Successor Condition). Node A, on processing a RREP rep, must relay the RREP only if $DInit_D^{rep}$= 1. Node A relays a RREQ for destination D only if A has not previously processed this RREQ, and sets $NRRL_D^{req} = \phi$.

USC: (Update Successor Condition). If node A must change s_D^A, then it sets $d_D^A \leftarrow \infty$, and issues a new RREQ req. If $src^{RRL_D^A}$= A, then it sets $NRRL_D^{req}$= RRL_D^A, else $NRRL_D^{req}$= ϕ.

USC allows a source to query its neighbors with its stored RRL to identify any loop-free paths to the destination. If no neighbor can answer the RREQ, then, as per RSC, the RREQ reaches the destination building a tree rooted at the source, and the RREP generated by the destination traverses one of the loop-free reverse paths along the tree. SSC allows neighbors to issue a reply that can be used at the source, and ASC allows the source to safely switch successors for a destination without causing any loops. If $NRRL_D^{req}$= ϕ, then the RREQ can only be answered by the destination.

2.1 Information Stored and Exchanged

The routing table at node A maintains the following parameters for every destination D: the successor (s^A_D), the route-request label (RRL^A_D), the distance (hop count) to the destination (d^A_D), lifetime of route, and the state of route entry (rt^A_D): valid or invalid. If no entry for destination D exists, then it is considered equivalent to having a *null* RRL (ϕ). Node A maintains a monotonically increasing source sequence number ID_A, which additionally serves as a route-request identifier.

A RREQ consists of the tuple $\{dst,\ src,\ rreqid,\ NRRL_{dst}{,}flags\}$. The field *src* denotes the identifier of the source that is seeking a path to the destination (*dst*), the flooding identifier *rreqid* along with the source (*src*) represents the unique route request label for this RREQ generated for a destination, $NRRL_{dst}$ is the neighbor-query RRL to find neighbors of the source that have a loop-free path to the destination. $flags$ carries control bits.

A RREP consists of the tuple $\{dst,\ src, rreqid,\ RRL_{dst},\ d_{dst}{,}ttl,\ flags\}$. The field *ttl* states the lifetime of the route at the node relaying the RREP, *rreqid* is carried in the RREP to forward it along the reverse path to the source using information cached for the RRL ($src, rreqid$), and also, if required, serves to form the new RRL, RRL_{dst} is the label stored at the relaying node for *dst*, d_{dst} is the distance to the destination at the relaying hop, and $flags$ contains the *'DInit'* bit, which is set when the destination originates the RREP (if set then RRL_{dst} is set to ϕ).

The RERR is the tuple $\{orig, unreachdests\}$, where *orig* denotes the node originating the route errors, and *unreachdests* is the list of destinations that are not reachable at *orig*.

A node relaying a RREQ identified by a RRL ($src, rreqid$) caches the address of the node *revhop* that sent the RREQ, and is used to relay a RREP received for this ($src, rreqid$) pair along the reverse path. Cached entries are maintained for a period of time that is long enough, so that all RREPs for the RREQ with RRL ($src, rreqid$) will be received.

2.2 Route Maintenance

(A) Initiating a RREQ: Node A is said to be *active* in a route computation for destination D (i.e., the RREQ) when it initiates a RREQ for the destination, and the RREQ is uniquely identified by the pair (A, ID_A). A node relaying a RREQ (A, ID_A) originated by another node is said to be *engaged* in the RREQ. A node that is not active or engaged in a route computation for destination D is said to be *passive* for that destination.

At any given time, a node can be the origin of at most one RREQ for the same destination. The RREQ (A, ID_A) terminates when either node A attains a successor for destination D or the timer for its RREQ expires.

If node A is active or engaged for destination D and receives data packets for the destination, it buffers those data packets. If node A is passive for destination D and requires a route for destination D, it sets $ID_A \leftarrow ID_A + 1$, $reqid \leftarrow ID_A$, and *RREQ timer* $\leftarrow$ *(*2.ttl.latency) (where *ttl* is the time-to-live of the broadcast flood and latency is the estimated per-hop latency of the network). If $src^{RRL^A_D} = A$, then $NRRL = RRL^A_D$, else $NRRL = \phi$. Node A then issues RREQ $\{D, A, reqid = ID_A, NRRL\}$.

If node A receives no RREP after the expiry of its timer for RREQ (A, ID_A) for destination D, it sends a new RREQ with an increased *ttl*. If node A does not receive a RREP for destination D after a number of attempts, a failure is reported to the upper layer. The number of hops that a RREQ can traverse is controlled externally from the RREQ by means of the TTL field of the IP packet in which a RREQ is encapsulated, or by other means.

(B) Relaying RREQs: When node B receives a RREQ $\{D,\ A,\ rreqid = ID_A,\ NRRL^{req}_D\ \}$ from node I, it first determines its own status for (A, ID_A). If B is active (i.e., $B = A$) or engaged (i.e., B has cached the RREQ (A, ID_A)) in the computation (A, ID_A), it silently drops the RREQ. Otherwise, node B is passive. In this case, if SSC is satisfied (i.e., $RRL^B_D \prec NRRL^{req}_D$), then node B issues a RREP (Section 2.2 (B)). Else, if SSC is not satisfied, node B becomes engaged and relays the new RREQ req' with $NRRL^{req'}_D \leftarrow \phi$.

A node may be engaged in multiple RREQs for the same destination, but relays a RREQ from the same origin only once by caching the reverse hop, $revHop = I$, for a given RREQ (A, ID_A) it forwards.

(C) Initiating and Processing RREPs: When node I processes a RREQ $\{D, A, rreqid = ID_A, NRRL^{req}_D\}$, if SSC is satisfied: $RRL^I_D \prec NRRL^{req}_D$, and $rt^A_D = valid$, it issues a RREP $\{D,\ A,\ ID_A,\ RRL^{rep}_D,\ d^{rep}_D,\ ttl\}$ with $RRL^{rep}_D \leftarrow RRL^I_D$ and $d^{rep}_D \leftarrow d^I_D$. At the destination $(I = D)$, D sets $DInit^{rep}_D \leftarrow 1$, $RRL^{rep}_D \leftarrow \phi$, and $d^{rep}_D \leftarrow 0$.

If node A receives a RREP $\{D,\ src = S,\ rreqid = ID_S,\ RRL^{rep}_D,\ ttl,\ d^{rep}_D,\ DInit\}$, it updates its routing table for destination D as described in Section 2.2(D). After updating its routing table, if $A \neq S$ and the RREP is not dropped, the node A increments ID_A, and the RREP is relayed along the reverse hop $revHop$ which is retrieved from the cache entry (A, ID_A). The RREP is relayed with the parameters $RRL^{rep}_D \leftarrow \phi$, $DInit^{rep}_D$ unchanged (must be one), and $d^{rep}_D \leftarrow d^A_D$.

(D) Adding, Updating, and Maintaining Routes: When node I receives RREP $\{D,\ A,\ ID_A,\ RRL^{rep}_D,\ ttl,\ d^{rep}_D,\ DInit^{rep}_D\}$ from neighbor B for destination D, it drops the RREP silently if ASC is not satisfied; otherwise, its routing table is updated as follows:

- Case (i), $DInit^{rep}_D = 1$,
  - if $I \neq A$, then node I sets $RRL^I_D \leftarrow [(A, ID^{rep}_A)]$.
  - if $I = A$, and $RRL^{rep}_D \prec RRL^I_D$, then node I retains the same RRL^A_D; otherwise, sets $RRL^I_D \leftarrow [(A, ID_I)]$.
- Case (ii), $DInit^{rep}_D = 0$, $I = A$, and $RRL^{rep}_D \prec RRL^A_D$ then node I(i.e., A) retains the same RRL^A_D.

In the next step, node I sets $s^I_D \leftarrow B$; and updates route cost, $d^I_D \leftarrow d^{rep}_D + lc^B_I$, where lc^I_B denotes the link cost from node I to B. When $DInit^{rep}_D = 0$, nodes only switch to shorter cost paths.

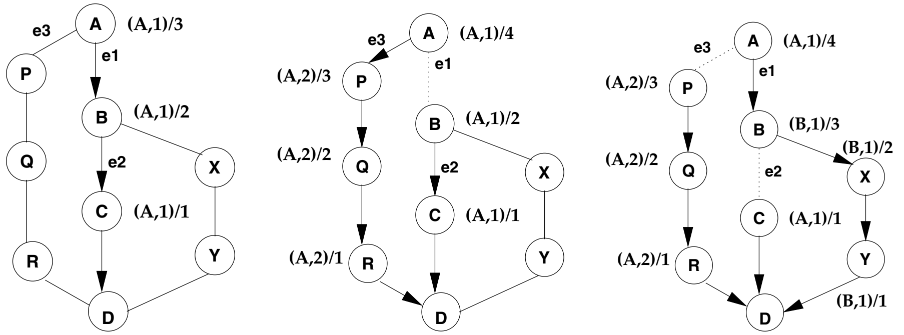
(E) Route Maintenance: Node A invalidates a route entry for destination D, with S as the next hop, in one of the following ways: (i) No data packet is forwarded using this route entry for *active_route_timeout* seconds (the time after which a route-entry expired); (ii) A link-failure notification for the next hop S is received; or (iii) A RERR is received, which indicates that D is no longer reachable through S. A node A invalidating an entry performs the following steps: It sets $rt_D^A = invalid$, $s_D^A \leftarrow \phi$, and $d_D^A \leftarrow \infty$. A route entry with state $rt_D^A = invalid$ can be purged at any time, to save memory (also, applies after a node reboot when nodes lose all state). For cases (ii), and (iii), node A sends a RERR to all the predecessors (either as a broadcast or separate unicasts), after determining the set of destinations affected by this event.

(F) Source Sequence Numbers: The source sequence number (ID_A) at a node A, serves the additional purpose of being the sequence number for route requests initiated by A. Because both purposes require a a monotonically increasing number (even after reboots), the ID_A must be based on a 64-bit real-time clock, which will avoid any wrap-around issues. Any repetition of the sequence number can cause route requests to be dropped at relay nodes because of previous cached state, and if old sequence numbers are repeated in RRLs then it can result in the formation of loops.

3 LSR Example

Figure 1 shows the directed acyclic successor graph (DASG) for destination D for a nine-node network at different instants of time. Link costs are unity. The figure shows, at each node for the destination D, a tuple $[(src, id)/hopcount]$, where (src, id) is the stored RRL, and *hopcount* is the distance to the destination. We illustrate the following sequence of events.

Node A has an active flow for destination D. Initially at time t_0, the routes are not active; and at all nodes, the RRL stored for D is ϕ. Node A initiates a route



(A) Time t_1 (B) Time t_2 (C) Time t_3

Fig. 1. LSR operation - Example

request req for destination D with parameters (D, A, $rreqid = ID_A = 1$, $NRRL = \phi$). Node B upon receiving the route request (A, $rreqid = 1$), caches the reverse hop ($revHop$= A), and relays a new RREQ req' with (D, A, $rreqid = 1$, $NRRL = \phi$). Similarly, node C relays the RREQ after caching reverse hop B. Note that the parameter $NRRL$ is set to ϕ because node B does not have an active route and the RREQ can subsequently be answered only by the destination.

A route reply rep is generated by node D upon receipt of the RREQ. The RREP (D, A, $rreqid = 1$, $RRL = \phi$, $DInit = 1$) is accepted by node C as it has $DInit$ set. Node C sets $s^C_D \leftarrow D$, $RRL^C_D \leftarrow (A, 1)$, $d^C_D \leftarrow 1$, and relays the RREP along the cached reverse hop. Similarly, because the RREP carries $DInit = 1$, nodes A and B switch successors and update their RRL and hopcounts for destination D as shown in Figure 1(A) at time t_1.

At time $t'_1 > t_1$, link $e1$ fails. Node A issues another RREQ with $ID_A = 2$, and sets up a new successor path along nodes P, Q, and R. Similar to the previous illustration, all the nodes switch successors and update their routing tables to set an RRL with $(A, 2)$ and the associated hopcount. Figure 1(B) shows the network state at time $t_2 > t'_1$. At any time later than $t'_2 > t_2$, if link $e3$ fails and link $e1$ comes back, node A can still switch successors to node B for destination D. This is because a RREQ carrying a $NRRL = (A, 1)$ will be answered by node B as it satisfies SSC, and because ASC is satisfied, node A can accept it. RRLs allow sources to switch to neighbors irrespective of any ordering based on distances (i.e., LDR) or path labels (i.e., FLR).

We illustrate loop-freedom in LSR with the following sequence of events after time $t''_2 > t'_2$. Assume node B becomes a source of data packets for destination D, and link $e2$ fails. Node B sends a RERR to A informing the loss of link to reach D. To recover from the failure of link $e2$, node B issues a new route request (D, B, $rreqid = 1$, $NRRL = \phi$), and on receiving a RREP from the destination with $DInit = 1$, labels the path along nodes X, and Y to destination D with $RRL = (B, 1)$. Assume at time $t'''_2 > t''_2$, link B-X fails. Regardless of whether node B's RERR was received by node A or not, new route requests from B cannot be answered by node A or any node upstream of it, and consequently loop-freedom is maintained even if RERRs cannot be reliably delivered. Figure 1(C) shows the network state at time $t_3 \geq t'''_2$.

Note that node A can still identify node C as a loop-free successor to destination D, because $RRL^C_D = (A, 1)$. Due to mobility, if node A moves closer to node Q or node R, it can still use them as successors to the destination. To summarize LSR's operation in a nutshell: After issuing a route request flood identified by a RRL, a source can identify all nodes which processed and relayed the RREP as loop-free successors to the destination. There are two cases after which this no longer applies: the source re-labels itself (increases to a higher sequence number in its RRL), say node A changed RRL to $(A, 3)$ at a later time; or the relay nodes processed a RREP carrying a RRL from a different source, as in the case of node B, here, which changed RRL from $(A, 1)$ to $(B, 1)$.

4 Analysis

We show that LSR is loop-free at any instant and can ensure that a source can establish a path to a destination in a stable, error-free connected network within a finite-time. We also show that in a connected component, separated from the destination, all nodes invalidate their routing table entries for the destination within finite time, guaranteeing termination.

Theorem 1. *LSR is loop-free at every instant.*

Proof. The proof is by contradiction. Let $P = \{A, ..., B, n_1, n_2, ..., I, ...D\}$ be a successor path along the directed acyclic successor graph (DASG) for destination D, which is loop-free at any time before t. At time t, let node I accept a RREP *rep* from node B, which is upstream to create a loop. We show that when nodes in LSR update their routing tables, it is not possible for I to switch successors to B. There are two cases by which node I will process the update: Case (i), the RREP from B has $DInit = 1$, which means the RREP was initiated by the destination, and must have a traveled a loop-free reverse path by virtue of the unique source tree built using the RREQ (src,id) pair (Theorem 2, [5], pp.49). Hence, I must have relayed the RREP onto B, and cannot receive the RREP again. This is a contradiction, and I can never switch successors to B. Case (ii), node I can switch to A, if $DInit_D^{rep} = 0$, and $RRL_D^{rep} = RRL_D^B \prec RRL_D^I$. Therefore, nodes I and B must possess RRLs, RRL_D^I =(I, ID'), and RRL_D^B= (I, ID''), respectively, such that $ID'' \geq ID'$. If $ID'' \geq ID'$, then it means that at a time $t^- < t$, node B must have relayed a RREP along a reverse path to I (which initiated the route request), and later switched to a node upstream of I. However, for node B to apply ASC to switch successors, it must have $src^{RRL_D^B}$= B which is not possible at time t^-. Hence, at a time $t^- < t^B < t$, node B could not have switched upstream of node I unless node B or one of its downstream successor path nodes to D was part of a RREQ flood (S, ID_S), creating a loop-free reverse path P'. If node I belonged to this path P', then $src^{RRL_D^I}(t_B)$= S, where $S \neq I$. Therefore, in this case, at time $t^B \leq t^I \leq t$, node I cannot apply ASC to switch successors to B. If at any time t^I $\leq t_+^I$ $<t$, node I adopted a RRL (I, ID'''), it must be true that $ID''' > ID''$ because ID_I is incremented on a RREP relay. Hence ASC will not be satisfied at time t, and no loops can be formed.

Theorem 2. *In a connected component G, partitioned from destination D, all nodes will invalidate their routing table entries for D within a finite time in the presence of link failures and node reboots.*

Proof. The one-hop neighbors of destination D, after partition, must detect the link failure to D within a finite time using a link layer notification scheme or otherwise (HELLO messages). Let t be the time after which the component G is partitioned from destination D and all RREPs initiated by the destination, with $DInit = 1$, have been processed by all nodes in G. Assuming default route error (RERR) message propagation from the neighbors of D along the directed acyclic successor graph (DASG) for destination D in the connected component G, we need to prove that nodes will not re-learn routes from any upstream nodes in the directed acyclic successor graph for D after time t. There are two states, that the nodes are in: Case (i), node A has a $src^{RRL_D^A} \neq A$, or $RRL_D^A = \phi$ which could also be because of a routing table state loss. By ASC, node A can update its routing table only if $DInit = 1$ in the RREP. But, because the destination D is partitioned, it is not possible to receive any new RREPs with $DInit$ set. Case (ii), node A has $src^{RRL_D^A} = A$, and on a link failure or otherwise, it can switch to successors that have $RRL \prec RRL_D^A$. However, as per the argument in Theorem 1, the nodes chosen as successors cannot be upstream of A in the DASG for D. Therefore, RERR messages should propagate along the DASG, hop-by-hop, within a finite time and all nodes should invalidate their routing entries in finite time.

Theorem 3. *In a stable, connected, error-free network, a source S can establish a route to destination D within a finite time.*

Proof. Let a source S start a route request req, identified by (S, ID_S), for a destination D. The RREQ must carry either (i) $NRRL = \phi$, or (ii) $NRRL = (S, ID'_S)$, where $ID'_S < ID_S$. In case (i), only the destination can answer the RREQ, and the RREQ must traverse a path $P = \{n_k, n_{k-1}, ..., n_1, D\}$ to reach the destination. Because the RREP will carry $DInit = 1$(i.e., initiated by destination), all the nodes will update their routing tables and relay the RREP along the reverse path of P. Therefore, S will establish a path to the destination D. In case (ii), there are two possible sub-cases: (a) a neighbor n_k of S that has an active route to D, can satisfy SSC. The RREP issued will satisfy ASC at S. Because of Theorem 1, there must exist a valid loop-free successor path to D from n_k because the network is error-free and connected. Hence, S will establish a route to the destination; (b) if neighbor n_k does not satisfy SSC, then it resets $NRRL = \phi$, which means only the destination can answer the request. Hence, this case follows directly from the same argument as in Case (i). In all cases, because the messages propagate in finite time, the source S can establish a loop-free path to the destination D in finite time.

5 Performance

We present results for LSR over varying loads and mobility. The protocols used for comparison are DSR, AODV, and OLSR which are very well known and being considered in the MANET working group of the IETF. Simulations are run in Qualnet 3.5.2. The parameters are set as in [8].

Simulations are performed on two scenarios, (i) a 50-node network with terrain dimensions of 1500m x 300m, and (ii) a 100-node network with terrain dimensions of 2200m x 600m. Traffic loads are CBR sources with a data packet size of 512 bytes. Load is varied by using 10 flows (at 4 packets per second) and 30 flows (at

4 packets per second). The MAC layer used is 802.11 with a transmission range of 275m and throughput 2 Mbps. The simulation is run for 900 seconds. Node velocity is set between 1 m/s and 20 m/s. Flows have a mean length of 100 seconds, distributed exponentially. Each combination (number of nodes, traffic flows, scenario, routing protocol and pause time) is repeated for nine trials using different random seeds. We present four metrics. *Delivery ratio* is the ratio of the packets delivered per client/server CBR flow. *Latency* is the end to end delay measured for the data packets reaching the server from the client. *Network load* is the total number of control packets (RREQ, RREP, RERR, Hello, TC etc) divided by the received data packets. *Data hops* is the number of hops traversed by each data packet (including initiating and forwarding) divided by the total received packets in the network. This metric takes into account packets dropped due to forwarding along incorrect paths. A larger value for the data-hops metric corresponding to a poor delivery ratio indicates that more data packets traverse more hops without reaching the destination necessarily.

Table 1. Performance average over all pause times for 50 nodes network for 10-flows and 30-flows

Protocol	Flows	Delivery Ratio	Latency (sec)	Net Load	Data Hops
LSR	10	0.9960±0.0016	0.0174±0.0022	0.3125±0.0797	2.5983±0.1813
AODV	10	0.9945±0.0023	0.0169±0.0033	0.2700±0.0668	2.5763±0.1793
DSR	10	0.9400±0.0274	0.0411±0.0477	0.2202±0.0952	2.6775±0.1853
OLSR	10	0.8870±0.0406	0.0129±0.0017	1.9370±0.2202	2.4568±0.1754
LSR	30	0.8358±0.0444	0.6081±0.2247	2.8147±0.8116	2.8338±0.2767
AODV	30	0.7651±0.0553	1.0101±0.3564	4.4233±1.2898	2.9514±0.3241
DSR	30	0.6837±0.0590	4.7600±1.0732	0.4108±0.1401	3.6253±0.3087
OLSR	30	0.7980±0.0349	0.8834±0.3113	0.7137±0.0695	2.4781±0.1618

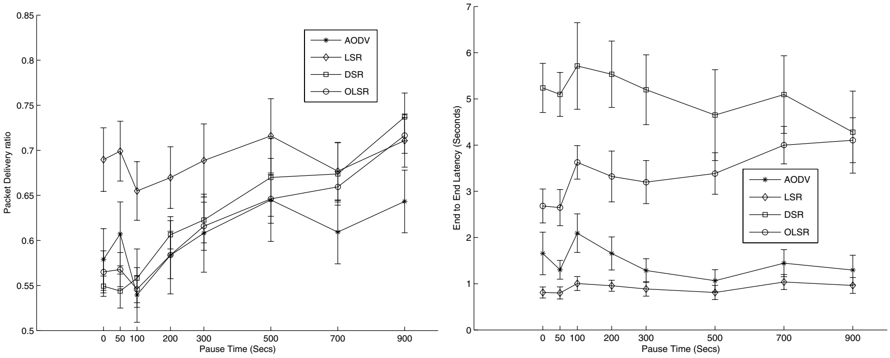
Table 2. Performance average over all pause times for 100 nodes network for 10-flows and 30-flows

Protocol	Flows	Delivery Ratio	Latency (sec)	Net Load	Data Hops
LSR	10	0.9910±0.0036	0.0383±0.0061	1.1618±0.3221	3.8182±0.3134
AODV	10	0.9882±0.0045	0.0366±0.0095	0.8973±0.2368	3.7441±0.2935
DSR	10	0.8760±0.0505	0.0993±0.0577	0.8599±0.3535	4.2573±0.3170
OLSR	10	0.8218±0.0637	0.0222±0.0020	11.7954±1.5754	3.5838±0.2567
LSR	30	0.6882±0.0356	0.9088±0.1508	10.8084±1.6577	4.4653±0.3530
AODV	30	0.6082±0.0517	1.4558±0.3856	18.2987±13.0698	4.7513±0.4340
DSR	30	0.6183±0.0496	5.1253±0.7820	1.2432±0.4053	6.1410±0.4999
OLSR	30	0.6126±0.0415	3.3714±0.5324	5.4231±0.6695	4.0142±0.2774

Tables 1, and 2 summarize the results of the different metrics by averaging over all pause times for the 50-node and 100-node networks. The columns show the mean value and 95% confidence interval. LSR has a very consistent performance across all scenarios and outperforms other protocols in most cases. Although the average results are statistically equivalent, the confidence intervals in most cases barely overlap.

In the highest load scenario (100 nodes, 30-flows), LSR has a packet delivery of 0.6882 ± 0.0356, and in the 50-nodes scenario with 30-flows, LSR's delivery ratio shares overlapping confidence intervals with AODV and OLSR. Figure 2(a) shows the delivery ratio for a 100-node network with 30-flows for different pause times. Confidence intervals (95%) are shown with vertical bars in the graphs. As reflected in the summarized average performance, LSR outperforms the other protocols by a wide margin in the high-mobility scenarios; DSR and OLSR have overlapping confidence intervals in the very low mobility scenarios.

In scenarios with 30-flows, LSR has a latency of 0.9088 ± 0.1508 seconds and 0.6081 ± 0.2247 seconds in the 100-nodes and 50-nodes scenario, respectively, better than the other routing protocols on the average. Figure 2(b) shows the data delivery latency for the 100-node network with 30-flows for different pause times. LSR has the lowest data packet latency at high mobility, and shares confidence intervals with AODV at low mobility.



(a) Delivery (100-nodes, 30-flow, 120pps) (b) Latency (100-nodes, 30-flow, 120pps)

Fig. 2. Performance results

The data hop count of all the protocols are comparable. Hence, LSR is delivering more data packets to destinations without dropping them due to congestion-triggered broken routes, and loops. The control overhead of LSR is better than that of AODV in all scenarios. DSR's and OLSR's control overhead cannot be directly compared since DSR uses the optimization to learn source-routes from data packets, and OLSR's control overhead is independent of the flows in the network.

6 Conclusion

We have shown that by using the same route request labels (RRL) needed as flood identifiers (i.e., a source address and an associated sequence number), it is possible to achieve loop-freedom even when nodes other than the destination reply. We presented a new on-demand loop-free routing protocol, the labeled successor routing (LSR) protocol, which labels each node processing and relaying a reply with the RRLs it processes. The source issuing the route-request flood can later identify the nodes that store the RRL as safe loop-free successors to the destination. Simulation results shows that LSR outperforms the on-demand and proactive routing protocols being addressed in the MANET Working Group of the IETF.

References

1. T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," Request for Comments 3626, October 2003.
2. H. Rangarajan and J.J. Garcia-Luna-Aceves, "Making On-demand Routing Protocols Based on Destination Sequence Numbers Robust," ICC 2005, Seoul, Korea, May 2005.
3. J. J. Garcia-Luna-Aceves and H. Rangarajan, " A New Framework for Loop-Free On-Demand Routing Using Destination Sequence Numbers", The 1st IEEE MASS, October 25-27, 2004, Fort Lauderdale, Florida, USA.
4. D. Johnson et al, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF Internet draft, draft-ietf-manet-dsr-10.txt, July 2004.
5. H. Rangarajan and J.J. Garcia-Luna-Aceves, "Using Labeled Paths for Loop-free On-Demand Routing in Ad Hoc Networks," *Proc. ACM MobiHoc 2004*, Tokyo, Japan, May 24–26, 2004.
6. C. Perkins et al., "Ad hoc On-Demand Distance Vector (AODV) Routing," Request for Comments 3561, July 2003.
7. I. Chakeres, et al.,"Dynamic MANET On-demand Routing Protocol (DYMO)," IETF Internet Draft, draft-ietf-manet-dymo-00.txt, February 2005.
8. C. Perkins et al. "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, 8(1):16 – 28, Feb 2001.

Penalty Shaper to Enforce Assured Service for TCP Flows

Emmanuel Lochin[1], Pascal Anelli[2], and Serge Fdida[1]

[1] LIP6 - Université Paris 6
[2] IREMIA - Université de la Réunion, France
{emmanuel.lochin, serge.fdida}@lip6.fr
pascal.anelli@univ-reunion.fr

Abstract. Many studies have explored the TCP throughput problem in DiffServ networks. Several new marking schemes have been proposed in order to solve this problem. Even if these marking schemes give good results in the case of per-flow conditioning, they need complex measurements. In this paper we propose a Penalty Shaper (PS) which is able to profile a set of TCP flows so as to improve its conformance to a desired target. The main novelty of this shaper is that the shaping applies a penalty delay which depends on the out-profile losses in a DiffServ network. This penalty shaping can be used with any classic conditioner such as a token bucket marker (TBM) or a time sliding window marker (TSWM). We made an evaluation of the Penalty Shaper on a real testbed and showed that the proposed scheme is easily deployable and allows for a set of TCP flows to achieve its target rate.

Keywords: Edge to Edge QoS, Assured Service, TCP.

1 Introduction

The Differentiated Services architecture [1] proposes a scalable means to deliver IP Quality of Service (QoS) based on handling of traffic aggregates. This architecture advocates packet tagging at the edge and lightweight forwarding routers in the core. Core devices perform only differentiated aggregate treatment based on the marking set by the edge devices. Edge devices in this architecture are responsibles for ensuring that user traffic conforms to traffic profiles. The service called Assured Service (AS) built on top of the AF PHB is designed for elastic flows. The minimum assured throughput is given according to a negotiated profile with the user. Such traffic is generated by adaptive applications. The throughput increases as long as there are available resources and decreases when a congestion occurs. The throughput of these flows in the assured service breaks up into two parts. First, a fixed part that corresponds to a minimum assured throughput. The packets of this part are marked like inadequate for loss (colored green or marked IN). Second, an elastic part which corresponds to an opportunist flow of packets (colored red or marked OUT). These packets are

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1108–1119, 2005.

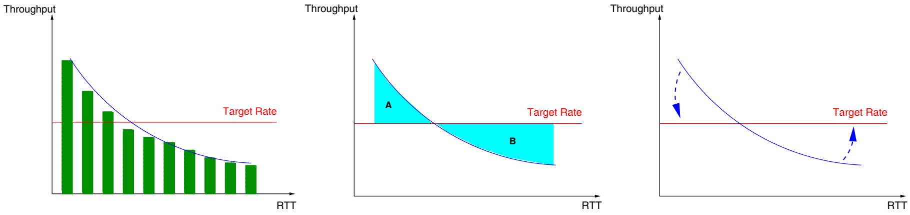
conveyed by the network on the principle of "best-effort" (BE). In the event of congestion, they will be dropped first. Thanks to an Penalty Shaper, we show that it is possible to provide service differentiation between two source domains, on a set of TCP flows, based on its marking profile. In this paper we evaluate the solution with long-lived TCP flows. The proposed solution provides the advantage of neither needing RTT(Round Trip Time) evaluation nor loss probability estimation. The solution takes care of the behavior of TCP flows only. Consequently, as it is easily deployable, it has been experimented on a real testbed. This paper is organized as follows. Section 2 presents related work. In section 2.1 we present the algorithm. Testbed and scenarios are presented in section 3. Section 4 presents the results obtained and their analysis. Finally, section 5 concludes the paper.

2 Related Work

There have been a number of studies that focused on assured service for TCP flows but also on the aggregate TCP performance. In [2], five factors have been studied (RTT, number of flows, target rate, packet size, non responsive flows) and their impact has been evaluated in providing a predictable service for TCP flows. In an over-provisioned network, target rates are achieved regardless of these five factors. This result is corroborated by [3]. However the distribution of the excess bandwidth depends on these five factors. When responsive TCP flows and non-responsive UDP flows share the same class of service, there is unfair bandwidth distribution and TCP flow throughputs are affected. The fair allocation of excess bandwidth can be achieved by giving different treatment to out-of-profile traffic of two types of flows [3]. Recently [4] demonstrates the unfair allocation of out-of-profile traffic and concludes that the aggregate that has the smaller/larger target rate occupies more/less bandwidth than its fair-share regardless of the subscription level. In [5], a fair allocation of excess bandwidth has been proposed based on a traffic conditioner. The behavior of the traffic conditioner has a great impact on the service level, in terms of bandwidth, obtained by TCP flows. Several markers have been proposed to improve throughput insurance [6, 7, 8, 9]. These algorithms propose to mark aggressive TCP flows severely out-of-profile so that they are preferentially dropped. Even if these marking strategies work well in simulation, their main disadvantage is their implementation complexity. Indeed, these algorithms need to measure a flow's RTT, its loss probability, have a per-state information of the flows or require a complex signaling algorithm.

2.1 The Penalty Shaper (PS)

Let $r(i)_{AS}$ be the assured rate of the flow i (*i.e.* in-profile packets throughput), n the number of AS TCP flows in the aggregate at the bottleneck level and C the link capacity. Precisely, this capacity corresponds to a bottleneck link in the network. If a number of i flows cross this link, the total capacity allocated for assured service R_{AS} is : $\sum_{i=1}^{n} r(i)_{AS}$. Let C_{AS} be the resource allocated to the assured service.



(a) (b) (c)

Fig. 1. TCP throughput of 10 TCP flows in function of the RTT

$$R_{AS} < C_{AS} \tag{1}$$

Equation (1) means an **under-subscription** network. In this case, there is excess bandwidth in the network. If $R_{AS} \geqslant C_{AS}$, this is an **over-subscription** network and there is no excess bandwidth. This configuration is the worst case for the AS. This service must provide an assurance until the over-subscription case is reached. Afterwards, not enough resources are available and the service is downgraded.

$$\mathit{TCP\ Throughput} = \frac{C * \text{Maximum Segment Size}}{RTT * \sqrt{p}} \tag{2}$$

The preferential dropping taking place at the core routers provides a good indication of the state of congestion. If the network is far from being congested, the in-profile packets will rarely be dropped and their dropping probability will be insignificant. If the network is going to be congested, almost all of the out-profile packets will be dropped. In a well-dimensioned network, inequity from (1) should be respected. When there are losses in the network, it corresponds to the losses of out-profile packets, and not in-profile packets. It means that a light congestion appears in the network and some out-profile packets must be dropped. In order to increase the loss probability of the opportunist flows, new conditioners presented in section 2 are based on increasing the out-profile part of the most aggressive traffic. Then, the loss probability raises and the TCP throughput of the opportunist traffic decreases. It's a logical behaviour because the latter has a reject probability higher than the non-opportunist traffic. [10] gives a model of TCP throughput represented by the equation (2). With C a constant and p the loss probability. Changing the p value from the equation (2) thanks to a marking strategy is complex. Indeed, it is necessary to evaluate the loss probability of the network and estimate an RTT for each flow. In order to illustrate this point, figure 1 presents the principle of these marking strategies. Figure 1 (a) symbolizes the throughput obtained by ten flows with different RTTs. The smaller the RTT, the higher is the throughput. The flows with a small RTT occupies more bandwidth than necessary as shown by area A in figure 1 (b). The aim of a marking strategy is to distribute fairly excess bandwidth from area A to area B. As a result, it corresponds to rotate the curve on figure 1 (c). As opposed to the marking strategy adopted by new conditioners, we propose a delay based shaper. This shaper applies a delay penalty to a flow if there are out-profile packets losses in the network and if it outperforms its target rate. The basic idea is that the penalty is a function of the out-profile packet losses. Instead of raising the p value, from equation (2), of the most opportunist flow, the Penalty Shaper raises a delay penalty to the flow. It results in a growth of the RTT. Mathematically, as shown in equation (2), increasing RTT value is similar to increasing p value in term of TCP throughput. [11] has shown that limiting out-profile packets is a good policy to achieve a target rate and a good solution to avoid TCP throughput fluctuations. Indeed, by avoiding packets dropping we avoid TCP retransmission. This is an efficient solution to optimize the bandwidth usage. That's the reason why we choose to operate on the raise of the RTT value instead of the raise of the p value. Moreover, we don't need complex measurements. Our goal is to reduce out-profile losses by applying a delay penalty to the flows that are the most opportunist in the network. Therefore, when a RIO[1] [12] router in the core network is dropping out-profile packets, it marks the ECN flag [13] of the in-profile packets enqueued in the RIO queue. In a well-dimensioned network, there is no in-profile packet loss. Then, the edge device can be aware that there is a minimum of one flow or set of flows which are opportunists in the network. This opportunist traffic is crossing the same path. The edge device evaluates its sending rate thanks to a Time Sliding Window (TSW) algorithm [14]. If its sending rate is higher than its target rate, it considers that its traffic may be opportunist. Then, it applies a penalty to the incoming traffic while the network feedback that there are out-profile packets losses. This penalty allows a raise of the RTT and consequently, decrease the TCP throughput. The algorithm presented in figure 2 explains how the penalty is calculated and applied. As explained on figure 4, once incoming TCP traffic is shaped, it passes through a marker such as a TBM. The PS mechanism could be placed on the client side rather than on the edge router. There isn't hypothesis on its localization. The PS is just used to enforce desired target rate and not used for marking traffic. In our simulation, the edge router uses a token bucket marker mechanism in order to mark in-profile and out-profile packets. Concerning the flows, traffic profile consists of a minimum throughput, characterized by two token bucket parameters, namely the token rate r and the size of the bucket b. The conformity control of an aggregate compared to the profile is thus done naturally by a token bucket as proposed in [15]. Figure 4 illustrates the ingress edge router mechanism.

The way TCP flows are conditioned by a marking strategy at the DiffServ network influences the drop probability of these TCP flows and consequently their behavior in the core network. Moreover, it is necessary to define a scalable conditioning in order to give an ISP[2] an exploitable solution. Many are

[1] RED with IN and OUT.

[2] Internet Service Provider.

```
K = 10 ms
T = 1 sec
FOR each observation period T
 TSW gives an evaluation of the throughput : throughput_measured
 IF throughput_measured < target_rate OR there are no out-profile losses
 THEN reduce the penalty delay
      current_penalty = current_penalty -  K
 ELSE raise the penalty delay
      current_penalty = current_penalty +  K
 END IF
END FOR
```

Fig. 2. PS algorithm

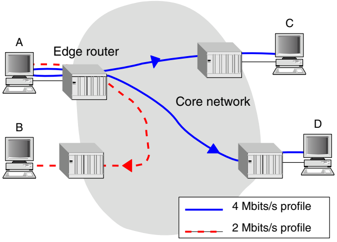


Fig. 3. Traffic conditioning sample

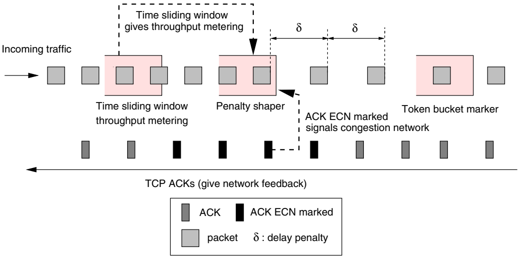


Fig. 4. Ingress edge router mechanism

the conditioners presented in section 2 which will never leave the framework of simulation because of their conditioning constraints. In order to avoid confusion with the aggregate principle defined in [1], in the remainder of this paper, we call TCP client aggregates a set of TCP flows emitted from one source to one destination. These aggregates can have one or several TCP flows. We chose to make the traffic conditioning in the following way : each client emitting one or more flows towards one or more destinations will have one traffic profile per destination. As shown on figure 3, client A forces the edge router to setup three different traffic conditioners. Two conditioners with a profile rate of $4Mbits/s$ and one conditioner with a profile rate of $2Mbits/s$. The main advantage of this solution is that the conditioning can be made on flows with similar RTTs (i.e. in the same order of magnitude). This solution doesn't depend on the complex problem of RTT estimation necessary to the functioning of the conditioners presented in section 2. The solution of traffic shaping coupled to a conditioner/marker such as the TBM should be easily deployable and scalable.

3 Experimental Testbed

As shown in figure 5, we use the well-known dumbbell topology. This topology was used in many experimentations [7, 11, 16, 17, 18]. The choice of this plat-

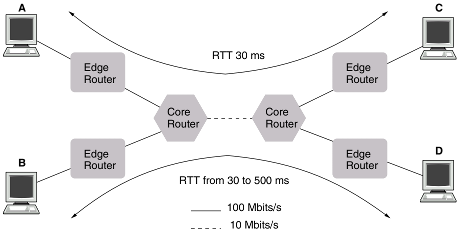


Fig. 5. Experimental testbed

form can appear simple, but it characterizes any router being in a core network. Whether it is on an edge router or on a core router, at the microscopic level, the behavior of each one is identical. At the macroscopic level, it is useless to serially chain several routers to carry out measurements because the end to end QoS will depend of the router under the worst conditions of traffic with regard to its output rate. Behind this router the flows are smoothed. Thus, a testbed made by a single router to build a bottleneck is sufficient to make the evaluation of the services proposed. The objective of this platform consist in evaluating the most significant QoS deviation. The testbed is composed of computers running Free-BSD. On the edge routers, the token bucket marker (TBM) from ALTQ[3]development and the Penalty Shaper based on Dummynet[4] On the core routers, a RIO queue, developed in ALTQ. We have added the marking functionality. Thus, the RIO queue is able to mark the ECN flag of the in-profile packets if it detects out-profile losses in its queue. Finally, we use two transmitting machines and two receivers for measurements. The acknowledgement management is important for reaching the targeted performance for TCP flows in DiffServ. The full-duplex link of the testbed allows forgetting TCP acknowledgment management. Indeed, no congestion occurs in the reverse path. The main parameters and hypothesis are : traffic generation is carried out in the following way: A to C (A, C) and B to D (B, D), after 120 seconds, `Iperf`[5] gives an average throughput of the flow ; each AS flow is transmitted as TCP, packets have a size of 1024 bytes ; `Iperf` uses a TCP maximum window size $Wmax = 64packets$; each set of flows between two hosts is conditionned by one TBM with or without PS ; b parameter of the TBM is set to one packet ; r parameter is set to the desired target rate ; the delay penalty is set to $10ms$ and the observation period to $1sec$. It means that each second the algorithm gives an estimation of the throughput and evaluates the penalty delay ; we use a non-overlapping RIO with parameters : $(min_{out}, max_{out}, p_{out}, min_{in}, max_{in}, p_{in}) = (1, 63, 0.1, 64, 128, 0.02)$, the queue size corresponds to $2 * Wmax$; we repeat each experiments five times and calculate the average throughput value (standard deviation is $\pm 0.15 Mbits/s$).

[3] `http://www.csl.sony.co.jp/person/kjc/`

[4] `http://info.iet.unipi.it/~luigi/ip_dummynet/`

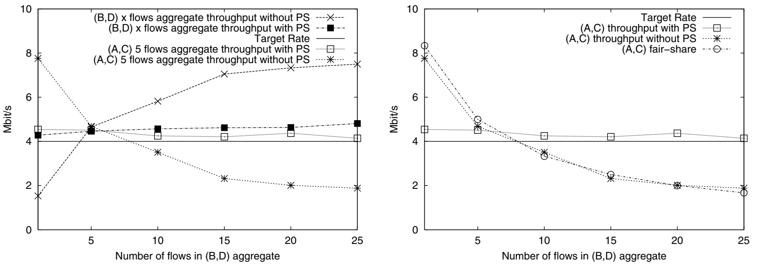
[5] `http://dast.nlanr.net/Projects/Iperf/`

4 Performance Evaluation of the Penalty Shaper

This section presents the results obtained in a real testbed with the PS. We evaluate the performance of the PS when client emitted TCP aggregate have the same or a different number of flows and identical or different RTTs. In all under-subscribed network scenarios : the total capacity allocated for the assured service is $R_{AS} = 8Mbits/s$ and the provisioned resource to the assured service is $C_{AS} = 10Mbits/s$ (that corresponds to the bottleneck capacity). So, there are $2Mbits/s$ of excess bandwidth.

4.1 Impact of the Aggregates' Aggressiveness in an Under-Subscribed Network

The main feature of the solution is : even if there is a different number of flows in the aggregates, the PS is able to reach its target rate. Results are presented in figure 6. When two aggregates with different number of flows are in a network, the higher outperforms the smaller. This aggregate aggressiveness problem was first raised in [2]. In these tests, two aggregates are in competition and the RTT of both aggregates is set to $30ms$. The (A, C) aggregate has a fixed number of 5 flows and the (B, D) aggregate has a variable number of flows ranging from 1 to 25. The target rate of both aggregates is set to $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. When (B, D) has less/more than 5 flows, (A, C) is the most/less aggressive aggregate. Figure 6 (a) shows the throughput obtained by both aggregates. The PS is able to reach the desired target rate. For clarification, we draw on figure 6 (b) the throughput obtained by the (A, C) aggregate alongside the fair-share curve. This figure shows that the TBM stays close to the fair-share while the PS is near the the desired target rate.

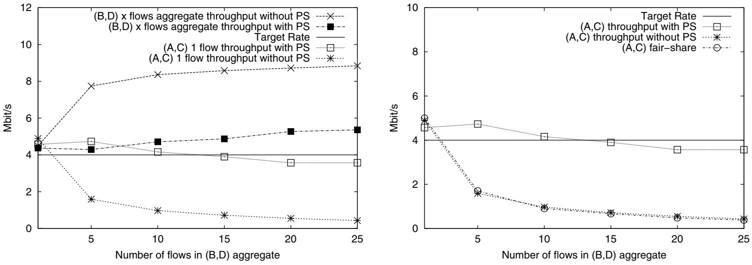


(a) (b)

Fig. 6. TCP throughput versus aggregates' aggressiveness

4.2 Impact of the Aggregate's Size on a Reference Flow in an Under-Subscribed Network

We use the same test conditions as previously but (A, C) is characterized by one flow only. We observe in this test the isolation of one reference flow in competition



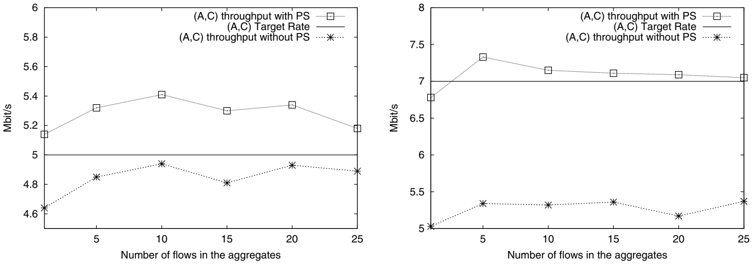
(a) (b)

Fig. 7. TCP throughput of one flow versus an aggregate

with an increasing number of microflows in the (B, D) aggregate. Figure 7 (a) shows that the PS obtains a throughput close to the desired target rate while the TBM stays always close to the fair-share curve as shown on figure 7 (b).

4.3 Impact of the Target Rate in an Under-Subscribed Network

In this section, two aggregates are in competition and the RTT of both aggregates is set to $30ms$. Both aggregates have the same number of flows. The number of flows varies from 1 to 25. Table 1 presents the results obtained in two scenarios. In the first one, (A, C) and (B, D) have respectively a target rate of $r(A, C)_{AS} = 5Mbits/s$ and $r(B, D)_{AS} = 3Mbits/s$ while in the second one, they have respectively a target rate of $r(A, C)_{AS} = 7Mbits/s$ and $r(B, D)_{AS} = 1Mbits/s$. So, the two scenarios illustrate both the case where the aggregates have near or distant target rates under under-subscription conditions. In figure 8 (a), we draw the throughput obtained by aggregate (A, C) when it requests a target rate of $r(A, C)_{AS} = 5Mbits/s$ (it corresponds to column 3 in



(a) (b)

Fig. 8. TCP throughput with various target rates

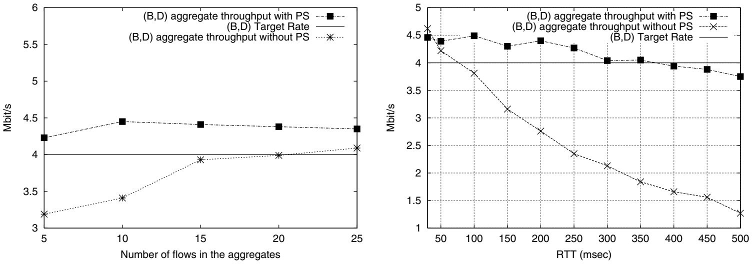
Table 1. Under-subscribed network (caption : goodput in $Mbits/s$ / target rate in $Mbits/s$)

Test	# flows aggregate	A to C	B to D	A to C	B to D
	(A, C) versus (B, D)	RTT=30ms	RTT=30ms	RTT=30ms	RTT=30ms
TBM only	1 vs 1	4.64/5	4.29/3	5.03/7	4.36/1
TBM+PS	1 vs 1	5.14/5	3.61/3	6.78/7	1.61/1
TBM only	5 vs 5	4.85/5	4.56/3	5.34/7	4.05/1
TBM+PS	5 vs 5	5.32/5	3.39/3	7.33/7	1.70/1
TBM only	10 vs 10	4.95/5	4.38/3	5.32/7	4.26/1
TBM+PS	10 vs 10	5.41/5	3.47/3	7.15/7	1.64/1
TBM only	15 vs 15	4.81/5	4.48/3	5.36/7	4.23/1
TBM+PS	15 vs 15	5.30/5	3.57/3	7.11/7	1.94/1
TBM only	20 vs 20	4.93/5	4.41/3	5.17/7	3.86/1
TBM+PS	20 vs 20	5.34/5	3.41/3	7.09/7	1.92/1
TBM only	25 vs 25	4.89/5	4.38/3	5.37/7	4.37/1
TBM+PS	25 vs 25	5.18/5	3.69/3	7.05/7	1.92/1

table 1) and figure 8 (b) shows the throughput obtained by aggregate (A, C) when it requests a target rate of $r(A, C)_{AS} = 7Mbits/s$ (it corresponds to column 5 in table 1). All the aggregates reach their target rate even if the number of flows in the aggregate increases.

4.4 Impact of the RTT in an Under-Subscribed Network

Even if there is a high number of flows in the aggregate and a high RTT difference, the PS is able to reach the target rate requested by an aggregate. Figure 9 illustrates this case. Figure 9 (a) shows the throughput of the (B, D) aggregate with an $RTT = 100ms$ versus an (A, C) aggregate with an $RTT = 30ms$, function of the number of flows. We focus on (B, D) aggregate throughput because in this scenario, (A, C) aggregate always reaches its target rate with or without the PS. The target rate for (A, C) and (B, D) is $r(A, C)_{AS} = r(B, D)_{AS} = 4Mbits/s$. Thanks to the Penalty Shaper, the (B, D) aggregate reaches the target rate and the increase of the number of flows in the aggregate doesn't influence the desired target rate. Finally, figure 9 (b)



(a) (b)

Fig. 9. TCP throughput versus RTT

shows the throughput of a 10 flows aggregate (B, D) in competition with a 10 flows aggregate (A, C). For the (A, C) aggregate, the RTT is equal to $30ms$ and for the (B, D) aggregate, we increase gradually the RTT from $30ms$ to $500ms$. It appears that the aggregate reaches the target rate when it is feasible (i.e. when target rate : $r(B, D)_{AS} > Wmax/RTT$). For information purposes, we did the same test with a number of flows ranging from 5 to 25 in both aggregates and obtained similar results. We present on figure 9 (b) only 10 versus 10 flows for space reasons.

4.5 A Case Study

In this part, we look at one (B, D) aggregate with five microflows in competition with three (A, C) aggregates constituted by ten microflows and one (A, C) UDP flow of $1Mbit/s$. Each (A, C) aggregate requests a target rate of $1Mbits/$ and has an $RTT = 30ms$. Table 2 gives the results obtained when the (B, D) aggregate has an RTT equal to $50ms$ or $100ms$ and requests a target rate of $4Mbit/s$. The (B, D) aggregate is in the worst conditions to reach its requested target rate but we can see that thanks to the PS, it obtains a TCP throughput near the target rate.

Table 2. Under-subscribed network (caption : goodput in $Mbits/s$ / target rate in $Mbits/s$)

Test	(B, D)'s RTT	TCP (B, D)	TCP #1 (A, C)	TCP #2 (A, C)	TCP #3 (A, C)	UDP (A, C)
TBM only	50ms	1.08/4	2.46/1	2.38/1	2.43/1	1.05/1
TBM+PS	50ms	3.71/4	1.40/1	1.41/1	1.43/1	1.05/1
TBM only	100ms	0.92/4	2.37/1	2.48/1	2.55/1	1.05/1
TBM+PS	100ms	3.65/4	1.36/1	1.35/1	1.42/1	1.05/1

4.6 Impact of the Number of Flows in an Over-Subscribed Network

In both scenarios in table 3, the total capacity allocated to the assured service is $R_{AS} = 12Mbits/s$. There is no excess bandwidth, the network is over-subscribed and there are several in-profile packets losses. Measurements in table 3 show that the PS allows to reach a TCP throughput closer to the target rate than a TBM. We don't discuss this case because we consider that it corresponds to a badly dimensioned service. Moreover, the PS algorithm, due to in-profile losses, is not really designed for this case.

5 Conclusion and Future Works

In this paper, we have studied on a real testbed a Penalty Shaper (PS) which provides throughput assurance between TCP flows. This is the first proposal that use a delay penalty which depends on the out-profile losses in a DiffServ network. The number of flows in the aggregate can influence the targeted throughput and balance the disadvantage of a long RTT flow or a difficult target rate. The main

Table 3. Over-subscribed network (caption : goodput in $Mbits/s$ / target rate in $Mbits/s$)

Test	# flows	A to C	B to D	A to C	B to D
	(A, C) versus (B, D)	RTT=30ms	RTT=30ms	RTT=30ms	RTT=30ms
TBM only	1 vs 1	4.46/8	4.50/4	4.94/10	4.31/2
TBM+PS	1 vs 1	4.69/8	4.33/4	6.40/10	2.31/2
TBM only	5 vs 5	5.02/8	4.34/4	5.25/10	4.05/2
TBM+PS	5 vs 5	5.08/8	4.21/4	7.01/10	2.09/2
TBM only	10 vs 10	5.08/8	4.25/4	5.24/10	4.06/2
TBM+PS	10 vs 10	5.38/8	3.98/4	7.20/10	2.07/2
TBM only	15 vs 15	5.02/8	4.31/4	5.27/10	4.08/2
TBM+PS	15 vs 15	5.53/8	3.74/4	7.32/10	2.05/2
TBM only	20 vs 20	5.05/8	4.36/4	5.43/10	3.84/2
TBM+PS	20 vs 20	5.51/8	3.73/4	7.29/10	2.09/2
TBM only	25 vs 25	4.90/8	4.35/4	5.29/10	3.95/2
TBM+PS	25 vs 25	5.49/8	3.79/4	7.34/10	2.04/2

consequence of these measurements is that we are able to obtain the guaranteed throughput if the profiled TCP aggregates in competition have the same or different number of flows. This is true whatever the differences between their RTTs and their target rates. In case of over-subscribed network, the PS is able to reach a throughput closer to the target rate than a simple token bucket marker. However, the PS is not designed for this type of network that corresponding to a badly dimensioned service. The proposed solution has the advantage of being easily deployable because it doesn't require complex measurements. The solution is scalable (it works at the edge of the network and is able to conditioning one or several flows) and being likely to be used with the most frequently used conditioners such as token bucket marker or time sliding window marker.

In our measurements, all the egde routers have setup the PS and all the core routers have a RIO ECN-capable queue. In a next work, we will analyse the case (with a large topology) where all the routers have not implemented our mechanism. Second, the penalty value is another critical issue. In our algorithm, this penalty is arbitrarily defined. We work on a penalty which follows the AIMD principle of TCP. So the penalty will be automatically calculated. We are currently deploying this proposal on a real large scale testbed with various traffic such long-lived and short-lived TCP flows.

References

1. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services. Request For Comments 2475, IETF (1998)
2. Seddigh, N., Nandy, B., Pieda, P.: Bandwidth assurance issues for TCP flows in a differentiated services network. In: Proc. of IEEE GLOBECOM, Rio De Janeiro, Brazil (1999) 6
3. Goyal, M., Durresi, A., Jain, R., Liu, C.: Effect of number of drop precedences in assured forwarding. In: Proc. of IEEE GLOBECOM. (1999) 188–193
4. Park, E.C., Choi, C.H.: Proportional bandwidth allocation in diffserv networks. In: Proc. of IEEE INFOCOM, Hong Kong (2004)

5. Alves, I., de Rezende, J.F., de Moraes, L.F.: Evaluating fairness in aggregated traffic marking. In: Proc. of IEEE GLOBECOM. (2000)
6. Kumar, K., Ananda, A., Jacob, L.: A memory based approach for a TCP-friendly traffic conditioner in diffserv networks. In: Proc. of the IEEE International Conference on Network Protocols - ICNP, Riverside, California, USA (2001)
7. El-Gendy, M., Shin, K.: Assured forwarding fairness using equation-based packet marking and packet separation. Computer Networks **41** (2002) 435–450
8. Feroz, A., Rao, A., Kalyanaraman, S.: A TCP-friendly traffic marker for IP differentiated services. In: Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS. (2000)
9. Habib, A., Bhargava, B., Fahmy, S.: A round trip time and time-out aware traffic conditioner for differentiated services networks. In: Proc. of the IEEE International Conference on Communications - ICC, New-York, USA (2002)
10. Floyd, S., Fall, K.: Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking **7** (1999) 458–472
11. Yeom, I., Reddy, N.: Realizing throughput guarantees in a differentiated services network. In: Proc. of IEEE International Conference on Multimedia Computing and Systems- ICMCS. Volume 2., Florence, Italy (1999) 372–376
12. Clark, D., Fang, W.: Explicit allocation of best effort packet delivery service. IEEE/ACM Transactions on Networking **6** (1998) 362–373
13. Ramakrishnan, K., Floyd, S., Black, D.: The addition of explicit congestion notification (ECN) to ip. Request For Comments 3168, IETF (2001)
14. Fang, W., Seddigh, N., AL.: A time sliding window three colour marker. Request For Comments 2859, IETF (2000)
15. Heinanen, J., Guerin, R.: A single rate three color marker. Request For Comments 2697, IETF (1999)
16. Mellia, M., Stoica, I., Zhang, H.: TCP-aware packet marking in networks with diffserv support. Computer Networks **42** (2003) 81–100
17. Nandy, B., P.Pieda, Ethridge, J.: Intelligent traffic conditioners for assured forwarding based differentiated services networks. In: IFIP High Performance Networking, Paris, France (2000)
18. Wang, F., Mohapatra, P., Bushmitch, D.: A random early demotion and promotion marker for assured services. IEEE Joural on Selected Areas in Communications **18** (2000)

QoS Dynamic Routing in Content Delivery Networks

Krzysztof Walkowiak

Chair of Systems and Computer Networks, Faculty of Electronics,
Wroclaw University of Technology, Wybrzeze Wyspianskiego 27,
50-370 Wroclaw, Poland
Krzysztof.Walkowiak@pwr.wroc.pl

Abstract. Recently, much research in quality of service (QoS) routing has focused on the unicast communication technique. However, Content Delivery Network (CDN) approach becomes popular because CDN enables effective and inexpensive improvement of Internet service quality. Therefore, we analyze in this paper a network processing two kinds of demands: content demands to CDN servers and standard unicast demands. Since MPLS defines effective mechanism for traffic engineering, we assume that the CDN is located in MPLS network. To examine the QoS performance of CDN-enabled MPLS network we propose new constraint-based algorithms for CDN server selection and evaluate relative performance of these algorithms and the most effective existing QoS unicast routing algorithms in terms of the demand rejection ratio.

Keywords: CDN, dynamic routing, QoS, server selection.

1 Introduction

Most of interest in context of Quality of Service (QoS) dynamic routing has been focused on the unicast traffic [1-2], [5], [7], [14], [17]. This is obvious, since unicast routing between a specified pair on network nodes is relatively well understood in best effort networks [4]. However, various techniques of network caching have gained much attention in recent years. In this work we focus on one of the most efficient caching approach – Content Delivery Network (CDN).

CDN is an interesting and robust approach to enhance the Internet quality. CDN uses many servers offering the same content replicated in various locations. User-perceived latency and the other QoS parameters (e.g. network reliability) can be easily and inexpensively improved by various techniques of Web content caching. Every replicated system must deal with: deciding on placement of replicas and distributing requests to object replicas. In this work we focus on the second problem. At present, conventional CDNs support only best-effort services. However, due to a big competition on the telecommunication market, Internet service providers will need to provide also QoS services to attract more clients [19]. We assume that the CDN is located in MPLS (Multiprotocol Label Switching) network. The MPLS approach proposed by the Internet Engineering Task Force (IETF) in [12] is a networking technique that enables traffic engineering (TE) for carrier networks. We consider a CDN system

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1120 – 1132, 2005.

with traffic engineering capabilities provided by the MPLS. Note that the traffic of CDN is also referred to as anycast traffic [4], [18]. The novelty of this work is that we analyze QoS performance of a network that can accept two kinds of demands: content demands to CDN servers and standard unicast demands. Only few previous works on this subject uses such approach [4]. There has not been, however, any study we are aware of that examines QoS dynamic routing of both unicast and anycast traffic in MPLS network.

The main goal of this work is to examine performance of various server selection algorithms and unicast dynamic routing algorithms in CDN network using the MPLS environment. Due to limited size of the paper, we don't present and discuss other issues related dynamic routing in CDN like: signalling protocols, client demand prediction, QoS network architecture, content distribution.

The remainder of the paper is structured as follows. Section 2 describes issues of CDN and MPLS. In Section 3, we present algorithms for content server selection. Section 4 includes results of simulations. We conclude in Section 5.

2 CDN-Enabled MPLS Network

Content Delivery Network is defined as mechanisms to deliver a range of content to end users on behalf of origin Web servers. The original information is offloaded from source sites to other content servers located in different locations in the network. For each request, the CDN tries to find the closest server offering the requested Web page [8]. CDN delivers the content from the origin server to the replicas that are much closer to end-users. The set of content stored in CDNs servers is selected carefully. Thus, the CDNs' servers can approach the hit ratio of 100%. It means that almost all request to replicated servers are satisfied [9].

Conventional CDN systems routes user requests to the nearest server using a redirection methods. Since redirection mechanisms don't directly select a route to the server, they are not effective in terms of network resources usage. Hence, some requests are rejected because the CDN redirects user to a server, which can be strongly loaded or links leading to this server are congested. Therefore, in this work we assume that the CDN consisting of content servers is located in MPLS network that provides traffic engineering capabilities. We call such a network CDN-enabled MPLS network. However, also other architectures can be used for provision of QoS and TE capabilities in CDN. For more details refer to [4], [18].

A user in the CDN-enabled MPLS network can generate two kinds of requests:

- **Unicast** is a standard MPLS unicast demand defined by a following triple: ingress router (source node), egress router (destination node) and the amount of required bandwidth requirement. A constraint-based dynamic routing algorithm is used to calculate a path for a unicast demand. If a feasible path doesn't exist, the demand is rejected.
- **Content** is a demand to the Content Delivery Network. It is defined by a triple: ingress router – node in which a CDN client is located, bandwidth requirement of upstream LSP (from the client to the server) and bandwidth requirement of down-

stream LSP (from the server to the client). When a content demand is issued, first QoS-based server selection algorithm is used to pick a content server. If none of CDN servers is reachable, the demand is rejected. Otherwise, two LSPs are established: upstream (from ingress node to server) and downstream in the opposite direction. A constraint-based dynamic routing algorithm is applied for calculation of both LSPs.

For simplicity, we assume that CDN servers provide the same long-lived content related to services like: distance learning, e-books, software distribution, archives of electronic entertainment (MP3 files, movies), FTP. Examples of unicast demands are: Voice over IP, teleconferences, exchanging of files, VPN, less popular WWW servers.

The CDN-enabled MPLS can deal with both kinds of demands in order to satisfy selected QoS constraints. It is worth mentioning that most of previous work on dynamic routing and CDNs consider only one of the two kinds of demands presented above. In our work we are interested in the combination of these two trends. It is much more realistic case, since usually existing network carry various kinds of traffic in the same links using the same routing algorithms.

One of the most important issues of CDN is the mechanism used for requests redirection. Transparent replication assumes redirecting a client's request for a document to one of the physical replicas. In [8] it has been presented the most popular practical and theoretical approaches of requests redirection: client multiplexing, IP multiplexing, DNS indirection, HTTP redirection and anycast, peer-to-peer routing. Generally, CDN routes user requests to the nearest or lowest-load server to reduce network latency. The effectiveness of a server selection strategy depends on its ability to take into account the metrics that contribute to the improvement of the QoS perceived by the users. Therefore, in our work we assume that when a user issues a new request to a CDN server, the special constraint-based server selection algorithm is applied. Such an algorithm uses the traffic engineering information on network saturation and selects the best content server in terms of network resources usage. We apply the explicit routing mode of MPLS [12]. Thus, the server selection can be transparent to end user, because the selection algorithm is located in the ingress node of MPLS network that is responsible also for the path selection. Similar approach referred to as source route option is mentioned in [18] in the context of anycast in Ipv6 networks. We assume that the following link state information applied for server selection either is flooded by the routing protocol or known administratively: total flow on link, link capacity, location of content servers and network topology. Three first items require extensions considering the traffic engineering provided by extended version of OSPF [6], [13].

3 Algorithms for Content Server Selection

In this section we present 3 algorithms that can be used for selection of content server when a content demand arrives in the network. The CDN-enabled MPLS network is modeled as (G,c), where $G=(V,A)$ is a directed graph with n vertices representing routers or switches and m arcs representing links, $c:A\rightarrow R^{+}$ is a function that defines capacities of the arcs. We denote by $o : A - V$ and $d : A - V$ functions defining the origin and destination node of each arc.

To mathematically represent the problem we introduce the following notations

f_a	Represents the total flow on arc *a*, it is calculated as a sum of all LSP's bandwidth requirements that uses arc *a*.
c_a	Capacity of arc *a*.
$r_a = c_a - f_a$	Residual capacity of arc *a*.
$g_v = \sum_{i:d(i)=v} f_i$	Flow of node *v* - aggregate flow of incoming arcs of *v*.
$e_v = \sum_{i:d(i)=v} c_i$	Capacity of node *v* - aggregate capacity of incoming arcs of *v*.
$h_v = e_v - g_v$	Residual capacity of node *v*.
Q_d^{up}	Upstream bandwidth requirement of demand *d*.
Q_d^{down}	Downstream bandwidth requirement of demand *d*.
o_d	Origin (ingress) node of content demand *d*.

We say that a content server is *located* in node *v* when it is connected to node *v* using a link of very high capacity. Consequently, the connection between content server and network node (e.g. router) cannot become a bottleneck. Furthermore, we make an assumption that every server can serve a large number of demands. Thus, we don't limit the amount of service that can be provided at any server. According to [9], it is a reasonable assumption, since increasing the number of content servers is much more difficult than increasing the capacity of a server. The number of servers is frequently given a priori due to cost and administrative reasons, while the capacity constraint can be overcome by adding more machines and storage space. Consequently, the only limitation of the content server workload is capacity of links leading the network node in which the server is located. However, server selection algorithms presented below can be easily modified to include also constraints on server workload or storage capacity.

In the server selection algorithms we use the *feasible* network approach applied in the context of QoS unicast routing algorithm [1], [5], [7], [14], [17]. The feasible network for a new demand consists of all routers and links, for which residual capacity exceeds bandwidth requirement of the new demand. More precisely, we consider bandwidth requirements of both paths: upstream and downstream associated with a particular anycast demand and use for calculation of feasible network the bigger value of these two bandwidth requirements. Thus, routing in the feasible network guarantees that acceptance of a new anycast demand will not violate the capacity constraint. Moreover, only content servers reachable in the feasible network can be taken into account in the selection what facilitates the process. Such servers are called *available*. Let B_d denote the set of servers that are available for demand *d*. Note that the terms server and server node are used interchangeably and they mean the node in which the server is located. Let $Q_d = \max(Q_d^{\mathrm{up}}, Q_d^{\mathrm{down}})$. To find the set B_d we do the following algorithm

1. **Prune the network.** Remove from the network each link for which $r_a \le Q_d$.
2. **Server checking.** Assign metric 1 to each link. For each node v with content server apply the shortest path algorithm (e.g. Dijkstra) to check if a path between o_d and v exists. If such a path exists add the node server v to the set B_d.

   Now we present three QoS server selection algorithms for a content demand d.

Hop Number Server (HNS) Algorithm

1. **Find available servers.** Calculate set B_d of available servers for demand d. If set B_d is empty, reject the demand and stop the algorithm.
2. **Find the server cost.** In the feasible network for demand d assign metric 1 to each link. For each server node v included in set B_d find the shortest path between v and o_d. Let $L(v)$ denote length of the path.
3. **Server selection.** Select server from the set B_d with the lowest value of $L(v)$. If two or more servers have the same minimal value of L, choose server for which residual capacity h_v of node v is the largest.

Hop Number Widest Server (HNWS) Algorithm

1. **Find available servers.** Calculate the set B_d of available servers for demand d. If set B_d is empty, reject the demand and stop the algorithm.
2. **Find the server cost.** In the feasible network for demand d assign metric 1 to each link. For each server node v included in set B_d find the shortest path between v and o_d. Let $L(v)$ denote length of the path.
3. **Server selection.** Select server v from the set B_d with the lowest value of $(L(v)/h_v)$.

Residual Capacity Server (RCS) Algorithm

1. **Find available servers.** Calculate the set B_d of available servers for demand d. If set B_d is empty, reject the demand and stop the algorithm.
2. **Server selection.** Select server from the set B_d with the largest value of residual capacity h_v of server node v.

In all algorithms we take into account only available servers. Server selection is done according to two metrics: hop distance to the server and the residual capacity of server node. The former metric ensures selection of the nearest (in terms of the hop number) server. However, this can lead to the congestion of network links leading to the server node. Therefore, the latter function is applied to balance the content demands among various servers in the network.

Computational cost of all three algorithms proposed above is O(mnr), where r denotes the number of content servers in the network. The major effort is to find shortest paths for all content servers. Therefore, complexity O(mn) of shortest path algorithm is multiplied by r. However, in some cases the number of servers is a constant and it does not depend on the network size (defined by the number of nodes). Consequently, under such assumption complexity of server selection algorithms can be reduced to O(mn) – the same as many dynamic online routing algorithms, e.g. Minimum Hop Algorithm (MHA), Constraint Shortest Path First (CSPF) [2], second stage of Dynamic Online Routing Algorithm (DORA) [14] and Least Interference Optimization Algorithm (LIOA) [1]. In contrast, Minimum Interference Routing Algorithm

(MIRA) [7] has complexity of (O(n^5)+O(m^2)), the first stage of DORA has complexity of O(n^3m^2).

Sometimes server selection algorithm may seem to overlap with the routing algorithm. However, one of our major assumptions is that the consider network supports and applies existing traffic engineering online routing algorithms (e.g. MHA, CSPF [2], DORA [14], LIOA [1]) and we do not have to change the routing protocol. Consequently, the same routing algorithm can be used for unicast and anycast demands. Therefore, we develop special algorithms for CDN server selection that cooperating together with unicast routing algorithms enable establishing of anycast demands. Nonetheless, it is possible to develop a mixed algorithm that can jointly select a server and find a path to the server. Such an approach leads to the situation, in which two various routing algorithms are indispensable in the network: one for unicast demands and one for anycast demands.

In this paper we focus on online, dynamic routing of demands. However, also many static routing optimization problems are widely discussed in the literature, e.g. [3], [10]. In [15-16] static versions of CDN network design problems are considered – heuristics and an exact algorithm based on the branch-and-cut approach are proposed.

4 Results

We now describe our simulation setup and scenarios. We conducted simulation experiments to evaluate the server selection algorithms: HNS, HNWS, RCS and four dynamic routing algorithms: MHA, CSPF [2], DORA using the bandwidth proportion parameter BWP=0.5 [14] and LIOA using the calibration parameter α=0.5 [1]. It makes 12 combinations of both types of algorithms. As the performance indicator we apply the demand rejection ratio. The objective of conducting performance evaluation is twofold. First, we want to evaluate various algorithms in terms of rejection ratio for various simulation scenarios. Next, we plan to examine the influence of number and location of CDN servers on the demand rejection.

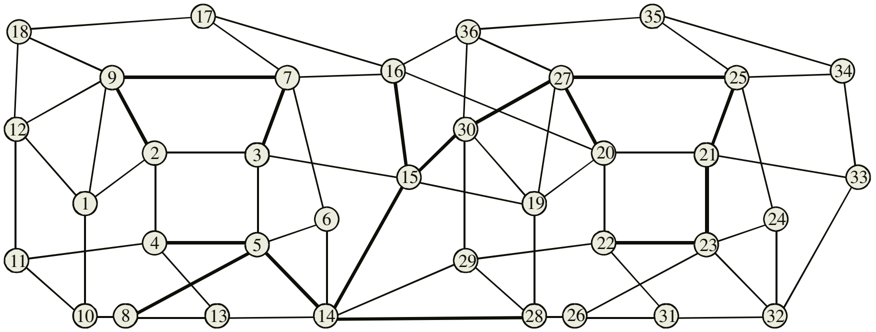


Fig. 1. Network topology used in the experiments

The network on which we conduct our experiment consists of 36 nodes and 144 directed links (Fig. 1). Each network node can be used as an ingress and egress node. The bold lines represent links of size 96 units while other lines are links of size 48 units. The link capacities were chosen to model capacity ratio of OC-48 circuits. During simulations, the link capacities were scaled by a factor 100 to enable establishing of thousands of LSPs. We consider static demands resembling long-lived MPLS tunnels that once established, they stay in the network for a long time. The same network was analyzed in [16].

In our experiments we conducted simulations for 12 combinations of content servers location (Table 1). The number of content servers is from 1 to 4. Since we want to examine how the number of content servers has an effect on demand rejection, we consider three cases (1A, 1B and 1C), in which there is only one server. Actually, we cannot call such scenario a CDN. Therefore, these three cases were not considered in the phase of algorithms comparison.

Table 1. Simulation scenarios of server number and server location

Case	Number of servers	Location of servers	Case	Number of servers	Location of servers
1A	1	14	3A	3	9, 14, 27
1B	1	15	3B	3	5, 23, 30
1C	1	30	3C	3	9, 15, 25
2A	2	14, 30	4A	4	5, 9, 23, 27
2B	2	5, 27	4B	4	5, 7, 25, 30
2C	2	9, 25	4C	4	9, 14, 23, 30

In the simulation we use the content demand ratio (CDR) parameter that is defined as the ratio of the content demands number to the number of all demands. For instance, CDR=0.3 means that 30% of all demands are addressed to content servers. In order to make performance evaluation for various scenarios, we generated randomly 10 sets of demands for each of the following five values of CDR: 0.1, 0.3, 0.5, 0.7 and 0.9. It gives 50 sets of demands in total. Each set consisting of 50000 demands was tested for each of 12 server location cases. It means that we have made 50x12=600 runs of the program that simulates combinations of server selection and routing algorithms. We assume that the requested bandwidth of unicast and content demand is randomly distributed between 1 to 10 units of bandwidth. However, in the context of content demand this is the downstream bandwidth requirement of the path from the server to the user. The upstream path in the opposite direction requires only 1 bandwidth unit. It is due to the asymmetric character of traffic in the Internet.

In the simulation we used the following methodology. For a content demand we first apply the server selection algorithm. If the algorithm cannot select a server, the demand is rejected. Otherwise, the dynamic routing algorithm is applied to find two LSPs between the client node and the server node and reserved bandwidth is updated on all links along both paths. A unicast demand is processed in a standard way, i.e.

the path between end nodes of the demand is calculated by a routing algorithm. The demand is rejected if the algorithm cannot yield a feasible path. Otherwise, reserved bandwidth is updated on all links along the path. Note that for both kinds of demand we apply the same routing algorithm. We repeat the same procedure for all 12 combinations of server selection and routing algorithms.

We do not model the real-time aspects of signaling protocols. We model these processes as functioning perfectly and we focus only on the purely capacity-related issues. After each individual demand placement, the available capacity is updated and the next demand in the sequence gets its chance, and so on until all demand have had a chance to be processed. Outcome of each experiment is the limiting outcome as it depends purely on basic routing and capacity considerations – which are repeatable by others. On the contrary, any attempt to also model the signaling dynamics is probably not repeatable as it is so utterly dependent on exact implementation details and simulation timing. Random timings and delays in signaling interactions with capacity seizure effects, protocol delays, and network timing effects will ultimately determine the result.

Table 2. Percentage of rejected demands aggregated over various scenarios

	Value of CDR parameter						Number of servers		
Algorithms	All	0.1	0.3	0.5	0.7	0.9	2	3	4
HNS_DORA	32.95	27.93	25.79	29.29	35.37	46.38	41.81	32.29	24.76
HNS_LIOA	**30.43**	**27.49**	**23.17**	**25.02**	**31.90**	**44.57**	**40.13**	29.64	**21.52**
HNS_CSPF	31.62	27.54	23.88	27.10	34.00	45.57	40.78	31.02	23.05
HNS_MHA	33.22	30.01	27.01	28.86	34.66	45.55	41.91	32.86	24.88
HWNS_DORA	33.08	27.98	25.99	29.47	35.50	46.45	41.89	32.38	24.97
HWNS_LIOA	30.44	27.49	23.22	**25.02**	**31.90**	**44.57**	40.14	**29.63**	21.55
HWNS_CSPF	31.62	27.51	23.90	27.11	34.00	45.58	40.79	31.01	23.06
HWNS_MHA	33.23	30.01	27.06	28.87	34.66	45.55	41.91	32.88	24.90
RCS_DORA	39.27	30.22	32.52	37.82	43.88	51.92	45.42	38.68	33.73
RCS_LIOA	36.58	29.53	28.69	34.13	41.29	49.24	43.46	35.50	30.69
RCS_CSPF	37.48	29.53	29.80	35.39	42.08	50.62	44.17	36.64	31.64
RCS_MHA	38.44	31.93	32.45	36.61	41.84	49.39	45.16	37.86	32.31

Table 2 shows the percentage of rejected demands for aggregated results. We present results aggregated for all tested values of CDR (second column) and individual results for each of CDR value (columns 3-7). Also the percentage of rejected demands obtained for different number of content servers placed in the network aggregated over all value of CDR is reported (columns 8-10). The best result (lowest rejection ratio) for each category is typed bold. The best performance is obtained for server selection algorithm HNS and routing algorithm LIOA. RCS algorithm yields higher rejection ration than two other server selection algorithm. Good performance of LIOA confirms the results presented in [1].

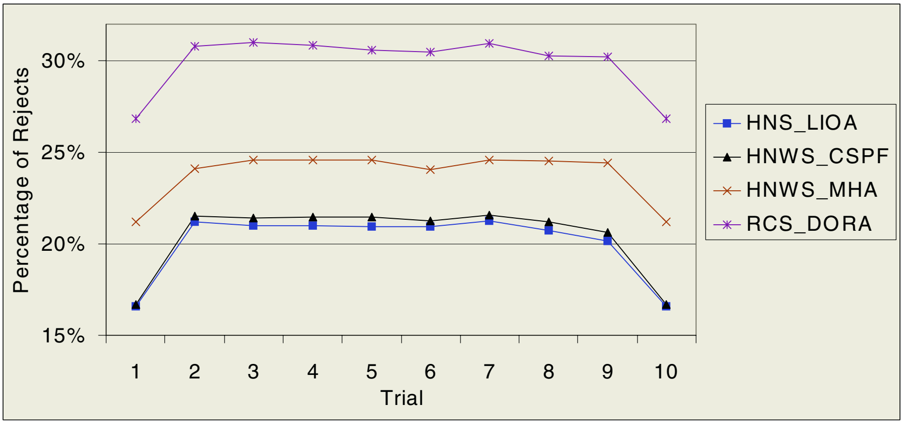


Fig. 2. Percentage of rejected demands versus trial number for case 4B and CDR=0.3

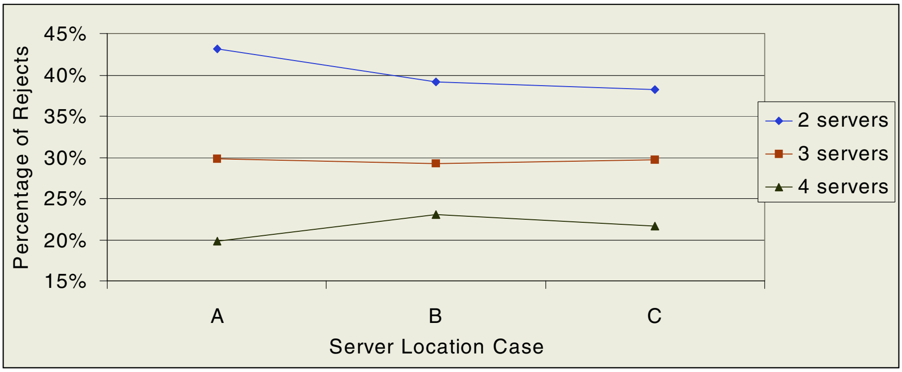


Fig. 3. Percentage of rejects for various simulation scenarios of location and number of servers

Figure 2 shows the percentage of rejected demands for different algorithms obtained for the case 4B with CDR=0.3. The general trend on Figure 2 is similar to that presented in Table 2. The comparison of various algorithms performance shows that combination of HNS and LIOA provide the best performance of demand rejects. Therefore, in the remainder of the section we present results of these two algorithms in order to examine the influence of the number and location of CDN servers on demand rejection ratio.

Figure 3 shows the percentage of rejects for various simulation scenarios in terms of the number of content servers and server location. Each point in the figure represents the result aggregated over all tested 50 demands patterns. As one can see, the performance depends on the server selection. The largest deviation is observed for 2 servers case. Obviously, when the number of servers increases, the rejection ratio goes down.

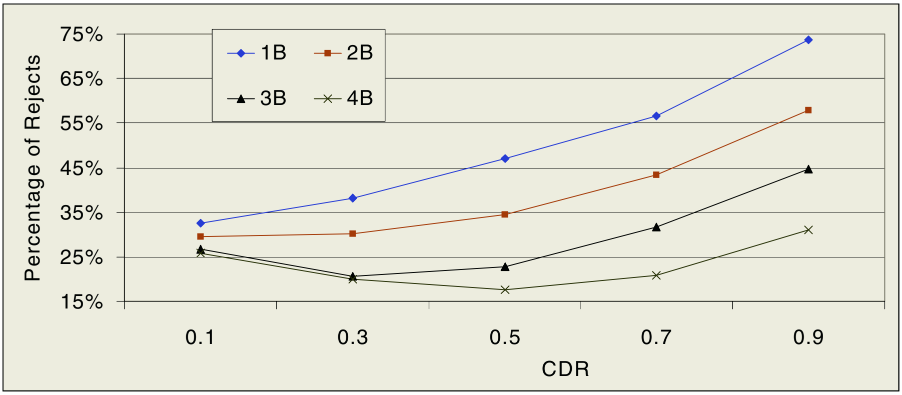


Fig. 4. Percentage of rejects as a function of CDR for cases 1B, 2B, 3B and 4B

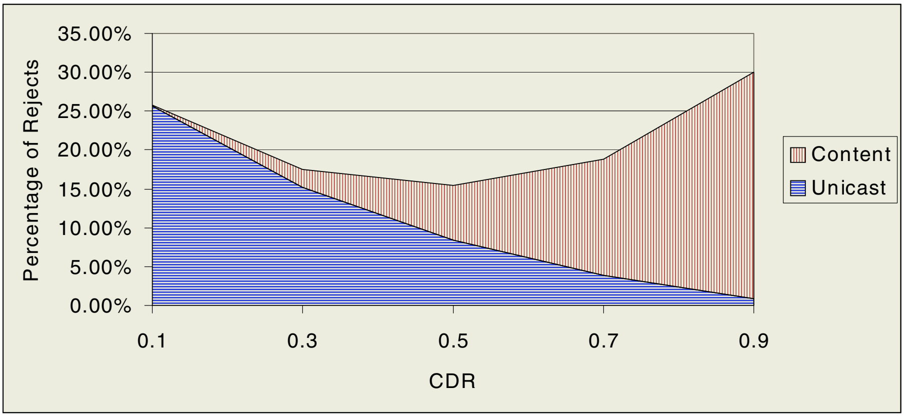


Fig. 5. Content vs. unicast demand rejection aggregated for 4 servers scenario

Figure 4 depicts the demand rejection as a function of CDR for scenarios 1B, 2B, 3B and 4B. If the number of servers is 1 or 2, the percentage of rejected demands grows with increasing of CDR. However, when the number of servers is 3 or 4, curves are different. Initially, when CDR is below 0.3 for 3 servers and 0.5 for 4 servers, the percentage of rejects decreases. It follows from the fact that there are more content servers in the network, which can accept content demands. Consequently, LSPs associated with content demands are shorter and leave more open capacity for other demands. However, when the ratio of content demands grows more, the limitation on capacity of links leading to servers surpasses the effect of higher number of content servers. Thus, relatively more content demands are rejected due to capacity constraint on links leading to server nodes. Similar trend can be observed on Figure 5 presenting the rejection ratio of content and unicast demands aggregated for all cases with 4 content servers.

Figure 6 shows the percentage of rejected demands as a function of demand number for cases 1B, 2B, 3B and 4B with CDR=0.5. We can watch that increasing the number of content servers reduces the number of rejects. For one server case after processing of 15000 demands the network starts to reject demands. If there are 4 content servers, about 37000 demands are accepted until demand rejection occurs.

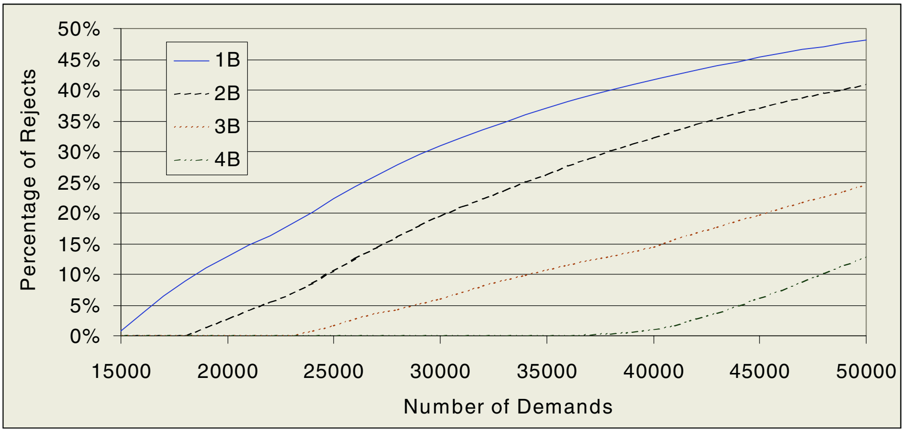


Fig. 6. Percentage of rejects as a function of demand number for cases 1B, 2B, 3B and 4B

5 Conclusion

In this paper we have examined a network that can accept two kinds of demands: content demands to CDN servers and standard unicast demands. We have proposed three algorithms for constraint-based selection of content servers. These algorithms either find the best server in terms of traffic engineering or reject the CDN request. One of the main advantages of our approach is that for routing of content demand the traditional unicast QoS dynamic routing algorithm can be applied. The only difference, comparing to unicast demands, is that two LSPs are established between client node and server node. We have studied the relative performance of four robust unicast routing algorithms cooperating with tree server selection algorithms. The lowest number of rejected demands provides the combination of HNS and LIOA algorithms.Our analysis of results suggests that both number and location of content servers contribute significantly to overall network reliability in terms of the rejection ratio.

Important factor that has en effect on the number of rejected demands is the proportion of content traffic.

As an ongoing work, we want to explore the performance of server selection algorithms in a CDN-enabled MPLS network from the perspective of network survivability. We plan to simulate content and unicast demand rerouting for various failure scenarios using different restoration methods.

Acknowledgements. This work was supported by a research project of the Polish State Committee for Scientific Research carried out in years 2005-2007.

References

1. Bagula, B., Botha, M., Krzesinski, A.: Online Traffic Engineering: The Least Interference Optimization Algorithm. To appear in the IEEE ICC 2004 (2004)
2. Crawley, E., Nair, R., Jajagopalan, B., Sandick, H.: A Framework for QoS-based Routing in the Internet. RFC2386 (1998)
3. Gola, M., Kasprzak, A.: Exact and Approximate Algorithms for Two–Criteria Topological Design Problem of WAN with Budget and Delay Constraints. Lectures Notes in Computer Science, LNCS 3045 (2004), 611-620
4. Hao, F., Zegura, E., Ammar, M.: QoS routing for anycast communications: motivation and an architecture for DiffServ networks. IEEE Communication Magazine, 6 (2002), 48-56
5. Kar, K., Kodialam, M., Lakshman, T.: Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. IEEE JSAC, 12 (2000), 2566-2579
6. Katz, D., Kompella, K., Yeung, D.: Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (2003)
7. Kodialam, M., Lakshman, T.: Minimum Interference Routing with Applications to MPLS Traffic Engineering. In Proceedings of INFOCOM (2000), 884-893
8. Krishnamurthy, B., Wills, C., Zhang, Y.: On the Use and Performance of Content Delivery Networks, in Proceedings of ACM SigComm Internet Measurement Workshop, (2001)
9. Peng, G.: CDN: Content Distribution Network. Technical Report, sunysb.edu/tr/rpe13.ps.gz, (2003)
10. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufman Publishers (2004)
11. Qiu, L., Padmanabhan, V., Voelker, G.: On the Placement of Web Server Replicas, in Proceedings of IEEE Infocom (2001), 1587-1596
12. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (2001)
13. Smit, H., Li, T.: ISIS Extensions for Traffic Engineering. RFC 3784 (2004)
14. Szeto, W., Boutaba, R., Iraqi, Y.: Dynamic Online Routing Algorithm for MPLS Traffic Engineering. Lectures Notes in Computer Science, LNCS 2345 (2002), 936-946
15. Walkowiak, K.: Some approaches to solve a Web replica location problem in MPLS networks, in Internet Technologies, Applications and Societal Impact, Kluwer (2002), 61-72,

16. Walkowiak, K.: An exact algorithm for design of content delivery networks in MPLS environment. Journal of Telecommunications and Information Technology 2 (2004), 13-22
17. Walkowiak, K.: Survivable Online Routing for MPLS Traffic Engineering. Lectures Notes in Computer Science, LNCS 3266 (2004), 288-297
18. Weber, S., Cheng, L.: A Survey of Anycast in IPV6 Networks. IEEE Comm. Magazine, 1 (2004), 127-132
19. Yamada, H. *et al.*: QoS Control by Traffic Engineering in Content Delivery Networks. Fujitsu Sci. Tech. J., 2 (2003) 244-254

Exploiting Traffic Localities for Efficient Flow State Lookup

Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao

Department of Computer Science and Software Engineering,
The University of Melbourne,
Victoria 3010, Australia
{tpeng, caleckie, rao}@cs.mu.oz.au

Abstract. Flow state tables are an essential component for improving the performance of packet classification in network security and traffic management. Generally, a hash table is used to store the state of each flow due to its fast lookup speed. However, hash table collisions can severely reduce the effectiveness of packet classification using a flow state table. In this paper, we propose three schemes to reduce hash collisions by exploiting the locality in traffic. Our experiments show that all our proposed schemes perform better than the standard practice of hashing with overflow chains. More importantly, our *move and insert to front* scheme is insensitive to the hash table size.

1 Introduction

The Internet is playing an increasingly important role in our society. A number of Internet applications, such as proxy services, firewalls and traffic billing need packet classification. In particular, due to the poor security level of the Internet, firewalls are an indispensable component for safeguarding on-line services. Due to the rapid increase in Internet bandwidth, processing speed has become a critical issue for firewall products. Although, firewalls based on FPGA hardware can provide high throughput, they are expensive and inconvenient to upgrade. A more flexible approach is to implement firewalls in software. The benefits of this solution are its low cost and ease of maintenance. However, high packet throughput becomes an issue. Consequently, the challenge is how to maintain high processing speed without compromising the sophistication of the firewall rule set. One important bottleneck in firewalls is caused by rule matching. In this paper, our aim is to exploit locality patterns in Internet traffic to increase the packet processing speed of firewalls.

A common technique to improve firewall performance is to use a flow state table. The firewall maintains a table of all the flows that it has authenticated. Before a packet is sent for rule checking, its flow information is checked to determine whether it appears in the flow state table. If the flow is in the table, the flow has already been seen, and so the packet is handled according to the classification in the table. If not, the new flow is sent for rule checking, and the

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1133–1145, 2005.

authenticated flow will be added into the flow state table. Generally, there are two types of data structures that are used to implement the flow state table, namely, hash tables and tree structures. Generally, hash tables are faster than tree structures for insertion and retrieval. A detailed comparison between hash tables and tree structures is outside the scope of this paper. In this paper, we focus on the performance of hash tables for implementing flow state tables.

Our contribution in this paper is to propose three hash table management schemes to improve the processing efficiency of flow state tables by exploiting traffic localities. The rest of the paper is organized as follows. In Section 2, we briefly review the use of hash tables for traffic flows. In Section 3, we propose three hash chain management schemes for handling the hash collisions. In Section 4, we use real-life packet traces to evaluate our schemes. In Section 5, we discuss other related issues for flow state tables.

2 Background on Flow State Tables

Internet services are carried via application sessions, such as downloading a web page or sending an email. Generally, one application session comprises several network sessions, such as TCP sessions or UDP sessions. All the packets involved in one network session belong to one IP flow. In this paper, unless otherwise stated, when we use the term flow we are referring to an IP flow. In IP v.4, we use 13 bytes as the flow identification, which includes the source address, source port number, destination address, destination port number, and protocol number, whereas in IP v.6 [4], we use the flow label and the source and the destination addresses. In this paper, we focus only on IP v.4, but the techniques developed here are also applicable to IP v.6.

We define the number of packets in each flow as the flow length. The flow length varies according to the type of application session. Some flows are short, such as a DNS lookup; some flows are long, such as transferring a large file via FTP. A common operation in network equipment is packet flow classification, for example, for security clearance and traffic management. With the rapid increase in network traffic speed, the computational overhead spent on packet classification has become a hurdle for achieving high packet throughput rates. Fortunately, all packets in the same flow share the same characteristics, such as integrity, priority and routing path. Hence, once one packet from a flow is classified, the rest of the packets within the same flow can share the same classification. This provides us with an opportunity to speed up the packet classification process by focusing only on flow classification.

A flow state table is one type of application that takes advantage of this feature to reduce per packet processing overhead. As shown in Figure 1, each incoming packet is checked to see whether it belongs to any of the flows in the flow state table. If it does, the packet is processed according to the decision associated with the matched flow. If it does not, the packet is classified by checking the whole rule set, and the classification result is used to update the flow state table.

As the overhead for flow state table lookup is much smaller than checking the whole rule set [5], the overall packet classification overhead is reduced. Gupta et al. gave an excellent tutorial on packet classification algorithms in [5]. However, to our best knowledge, no research has been done at the time of writing this paper on reducing the cost of flow state table lookup.

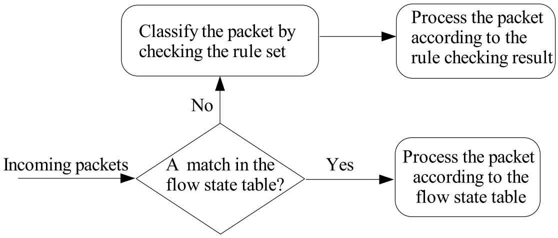

Classify the packet by checking the rule set
Process the packet according to the rule checking result
No
Incoming packets
A match in the flow state table?
Yes
Process the packet according to the flow state table

Fig. 1. Flow state table and packet classification

Several data structures have been proposed for flow state table lookup, e.g., search trees and hash tables [6] [3]. Typically, hash tables have the advantage of fast lookup (O(1) if no hash collision happens), while search trees have the advantage of dynamically adjusting the size of the data structure to the amount of data that needs to be stored.

Our focus is on using a hash table to store the flow state information. Typically, the 13-byte flow information f is first reduced using a hash function $h(f)$, which returns an index into the hash table. If the hash table has N entries, then the hash function returns an index in the range $0 \leq h(f) < N$. In practice, multiple flows can have the same hash index. Consequently, each entry in the hash table, known as a *bucket*, is a pointer to a linked list of flow records that have all been hashed to that entry. When two different flows map into the same bucket, a *collision* has occurred. These linked lists are also referred as *hash chains.* A critical issue for the performance of insertion and retrieval is how to manage records in the linked list.

For example, one conventional approach taken by a well-known open source firewall package IPfilter [6] is to always insert the new flow state record at the front of the linked list. The idea behind this approach is that once the new flow state is inserted, it is likely to be followed by packets from the same flow. Hence, most of the packets will match the first node of the hash chain. Unfortunately, tens of thousands of flows can be active at the same time in practice. In particular, several flows that are mapped into the same hash bucket can be active at the same time. Generally, these flows interleave with each other. For example, let A_i represent one packet from flow A, B_i represent one packet from flow B, C_i represent one packet from flow C, and flows A, B, C are mapped into the same hash bucket. For IP filter, the ideal sequence of the three incoming flows will be $A_1A_2A_3A_4A_5...B_1B_2B_3B_4B_5...C_1C_2C_3C_4C_5....$ In practice, the sequence is more likely to be interleaved, e.g., $A_1B_1A_2A_3A_4B_2B_3B_4C_1C_2A_5B_5C_3C_4C_5....$

In this scenario, the traffic locality cannot be fully exploited by just inserting the new node at the front of the list.

Let us consider another extreme example. Assume four active flows are mapped into the same hash bucket. All these four flows are active for the same time period. Three of them are slow flows, e.g., ssh sessions, and one of them is a fast flow, e.g., a large file download. We assume the fast flow starts *before* the three slow flows. Once these four flows are included in the *hash chain*, their positions in the hash chain will never be changed given no new flows are mapped into the same hash bucket. As shown in Figure 2, the fast flow who has the dominant number of packets of these four flows will have to search to the end of the list to find a matching flow. In this scenario, most of the packets will experience the longest searching time. Consequently, due to the complicated packet arrival sequence, simply inserting the new flow state at the front of the list is not enough to guarantee fast lookup performance. Better algorithms are needed to carefully exploit locality in the traffic flows.

3 Proposed Schemes for Flow State Table Management

Our aim is to reduce the time needed for flow state table lookup, i.e., hash table lookup, so that the overall packet processing speed is increased. The key step to improving the hash table lookup speed is to organize the hash chain so that most of the packets only need a few searches to find a matching flow. As we discussed in Section 2, we need a more adaptive scheme to exploit traffic locality. Our assumption is that for each hash bucket, the packets from the same flow will arrive continuously for at least two packets. We call these continuously arriving packets from the same flow a *burst*. For example, the sequence $A_1B_1A_2A_3A_4B_2B_3B_4C_1C_2A_5B_5C_3C_4C_5$... contains several bursts, such as $A_2A_3A_4$ and $B_2B_3B_4$. The conventional approach only focuses on organizing the hash chain according to the first packet of each flow, and fails to exploit the locality within each burst. To address this shortcoming and inspired by Zobel's idea in accumulating text vocabularies [7], we propose three different algorithms focusing on exploiting the locality within a burst.

– *Move to front*: This scheme maintains a linear list, and moves the most recently matched node to the front of the list, and inserts new flows at the end of the list.
– *Insert and move to front*: This scheme also maintains a linear list. However, it not only moves the matched flow to the front but also inserts the new flow to the front.
– *Circular list*: This scheme maintains a circular list for each hash bucket, and the hash bucket either points to the last matched flow or the newly inserted flow as shown in Figure 3.

All of these three schemes share the same feature that each matched flow will trigger a restructure of the hash chain, i.e., moving the matched flow to

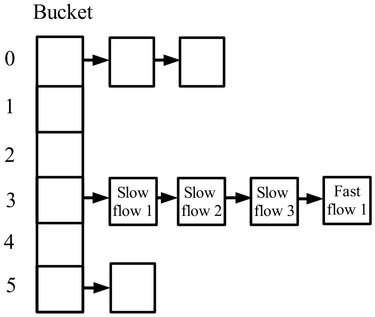


Fig. 2. Worst-case scenario for the conventional approach to managing the hash chain

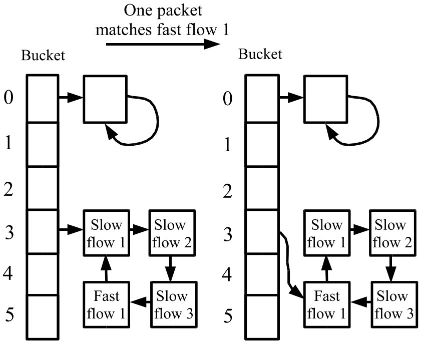


Fig. 3. The circular list hash chain

the beginning of the list. Hence, at least the second packet of each burst will only need one search operation to find a matched flow. In the scenario shown in Figure 2, once one packet from the fast flow 1 searches till the end of the hash chain to find a match, the fast flow 1 will be moved to the front. Hence, the following packets from the fast flow 1 only need one search to find the matching flow. For the simplicity of comparison, we refer to the approach taken in IP filter as the *normal* scheme, which is described in detail in Section 2 as the conventional approach. In the next section, we will use real-life data traces to verify our heuristics.

4 Evaluation

Our aim is to use real-life packet traces to validate our proposed schemes. There are two sets of packet traces that we use in the evaluation. The first packet trace is collected at the edge router of the Department of Computer Science and Software Engineering of the University of Melbourne in Australia. We refer to these as the *Melbourne traces*. The advantage for this packet trace is that it is recent and the whole IP packet header information is preserved. Unfortunately, these packet traces are not publicly available yet due to the University privacy policy. The second packet trace is collected at the Internet uplink of the University of Auckland in New Zealand [1]. We call these the *Auckland traces*. The Auckland traces are publicly available so that our results on this data trace can be easily reproduced by other researchers. The details of these two data sets are shown in Table 1.

Our experiments include on-line evaluation and off-line evaluation. For the Melbourne trace, we conducted an on-line evaluation. In the flow state lookup engine we implemented the normal scheme and our three proposed flow state table lookup schemes. In the traffic generator we use tcpreplay [2], an open source tool, to replay the traffic traces collected. Both the flow state lookup engine and the traffic generator comprise a 2.8 GHz Pentium 4 processor with

Table 1. Data sets used in evaluation

	Melbourne traces	Auckland traces
Direction	bi-directional	uni-directional
Date of Collection	22 May 2004	29 March 2001
Duration	24 hours	24 hours
Total Number of Packets	153,399,408	43,226,495
Format	tcpdump format	DAG format

1 MB L2 cache and 512 MB RAM. For the Auckland trace, we conducted an off-line evaluation [1] in a linux PC with dual 900MHz Xeon CPUs (each CPU has 512 KB L2 cache), and 512 MB RAM. Instead of reading the traffic from the network card, our programs read the traffic from the data trace file. As the total number of flows in each data set is large and we only need to maintain the states for the active flows, we need a mechanism to delete the old flow states and add the new flow states. For the convenience of evaluation of each hash chain management scheme, we simply remove the last 5 flows states once the hash bucket size reaches 10. The key goal of our experiments is to investigate how our algorithm improves the flow state lookup efficiency. The hash function we use is the bit-wise hash function from Zobel et al. [7].

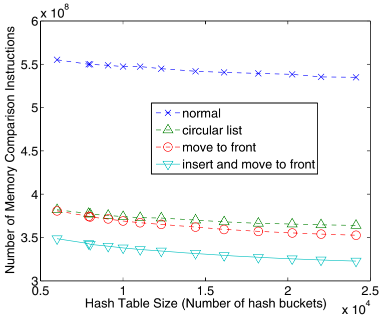


Fig. 4. Number of memory comparison instructions used by each scheme for IP traffic in the Melbourne trace

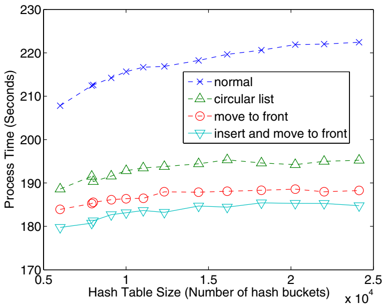


Fig. 5. Processor time used by each scheme for IP traffic in the Melbourne traffic

4.1 Process Efficiency

We use two different metrics to evaluate the efficiency of each flow state lookup scheme. First, we count the number of *memory comparison instructions* each

[1] The data server that was used to store the Auckland traces did not have enough resources to conduct an on-line evaluation.

scheme has used. Note that for simplicity, we count the number of memory comparison instructions rather than the total number of instructions executed. In our experiment, we count the number of calls to *memcmp()* used by each scheme. A smaller number of instructions indicates that the traffic localities have been better exploited, and the scheme has high efficiency. Second, we record the time spent by the processor (not the elapsed time) to process the same amount of traffic for each scheme. Each scheme only processes the IP traffic. Moreover, we vary the hash table size to see how this affects the performance of each scheme.

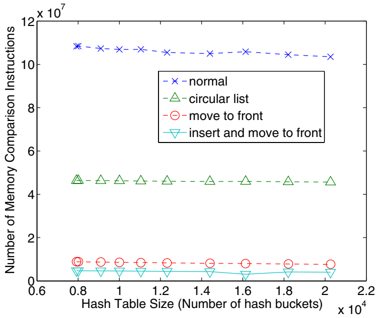


Fig. 6. Number of memory comparison instructions used by each scheme for IP traffic in the Auckland trace

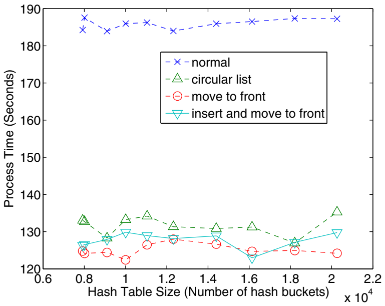


Fig. 7. The process time used by each scheme for IP traffic in the Auckland trace

Figure 4 illustrates the performance of our schemes for the Auckland traces. The *normal* scheme used 1.1×10^8 memory comparison instructions, the *circular list* scheme used 4.6×10^7 memory comparison instructions, while the *move to front* scheme and the *insert and move to front* scheme used only 8.4×10^6 and 4.5×10^6 memory comparison instructions respectively. Similarly, Figure 6 illustrates the performance of our schemes for the Melbourne traces. The evaluation results are quite similar to the Auckland traces. The *insert and move to front* scheme still performs the best, followed by the *move to front* scheme and *circular list* scheme.

The *move and insert to front* scheme performs the best because it keeps the more recent flow closer to the front of the hash chain. As the *move to front* scheme inserts the new node at the end of the hash chain, the entire hash chain has to be searched to find a matched node when the next packet of the new flow arrives. Hence, the *move to front* scheme needs more memory comparison instructions than the *insert and move to front* scheme. As the *circular list* leaves the pointer at the most recently matched node, the node in front can become the last node. As shown in Figure 3, if an incoming packet matches the slow flow 2, the hash bucket leaves the pointer at the slow flow 2. All packets from slow flow 2 only need one search to find a matching flow. However, if the next incoming packet is from slow flow 1, then it needs 4 searches to find the matching flow entry. This problem can be exacerbated if we have multiple fast flows in the same bucket. Consequently, the number of memory comparison instructions needed by the *circular list* scheme is larger than both the *move to front* scheme and the *insert and move to front* scheme. Figure 5 illustrates the performance of all four schemes in terms of processing time. We can see that the *move to front* scheme, the *insert and move to front* scheme, and the *circular list* scheme have very similar performance, which is about 30% faster than the normal scheme. The benefit for the *circular list* scheme is that only one operation is needed to let the hash bucket point to the recently matched node, which is simpler than the linear list. This benefit is diminished by the disadvantage of a large number of searches. Consequently, the overall processing time for the *circular list* scheme is very close to the *move to front* scheme and the *insert and move to front* scheme. Similar results are shown in Figure 7. We can see that in the Melbourne trace the *insert and move to front* scheme clearly stands out as the best scheme. More importantly, for hardware implementation, the hash chain can be implemented as a ring buffer instead of a linked list, where the size of the bucket is fixed. In this scenario, the *circular list* is more efficient than the other three schemes as it only needs to keep track of the most recently matched flow, and does not need to keep the rest of the flow entries in order.

4.2 Traffic Type

In some flow state table lookup implementations, e.g. a firewall application, the TCP, UDP, and ICMP flows are treated differently. For example, firewalls will have different sets of rules for each protocol. More importantly, only TCP traffic requires an initial handshake to establish a connection, and is regarded as stateful traffic. Hence, maintaining a flow state table for TCP traffic is essential for many Internet applications. This section investigates how our schemes perform under TCP traffic. Figures 8 and 9 illustrate the results for the Auckland traces. Figures 10 and 11 illustrate the results for the Melbourne traces. We can see that all of our three schemes perform better than the normal scheme. Overall, the *insert and move to front* scheme performs the best.

4.3 Hash Table Size

In this section, we aim to investigate the effects of the hash table size on the performance of each scheme. As we see from the Figures 4 to 11, the flow state table lookup performance is not sensitive to the hash table size. In theory, a larger hash table size can reduce the hash collision rate and the average hash chain length, and hence enhance the flow state table lookup performance. However, in our experiments, the hash chain length is reduced to be 5 once it reaches 10 in order to delete old flows. As the total number of flows is much larger than the number of flows the hash table can hold, each hash chain length is likely to reach 10 and will be reduced to 5 afterward. Hence, the average hash chain length is always between 5 and 10 for hash tables with different sizes. The benefit of a large hash table is to hold more flow states and reduce the number of flows that

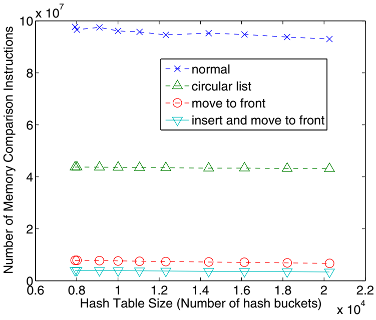


Fig. 8. Number of memory comparison instructions used by each scheme for TCP traffic in the Auckland traces

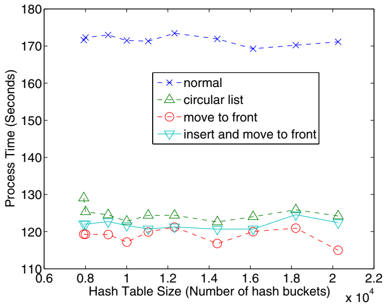


Fig. 9. Processor time used by each scheme for TCP traffic in the Auckland traces

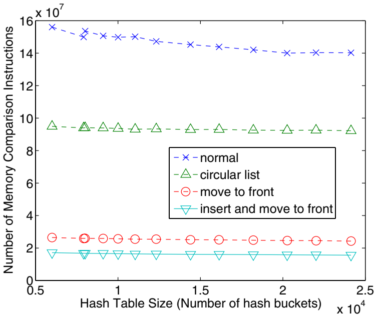


Fig. 10. Number of memory comparison instructions used by each scheme for TCP traffic in the Melbourne trace

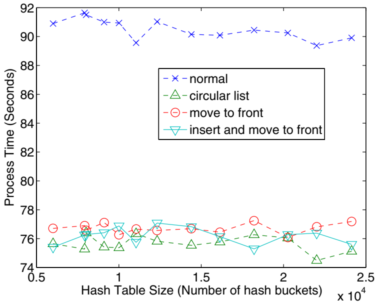


Fig. 11. Processor time used by each scheme for TCP traffic in the Melbourne trace

need to be re-inserted after deletion. The major cost of inserting a new flow is to classify the flow, i.e., deciding whether to admit or reject the flow. This cost is highly application dependent. Consequently, we do not consider this cost in our experiments, and a flow is inserted directly once no matching flow is found. For this reason, the hash table size has little effect on the flow state table lookup performance.

To facilitate our study of the impact of the hash table size, we performed an extreme experiment, where no flows are purged from the hash bucket. However, this experiment is limited by hardware, i.e., the RAM size. We conducted this experiment on TCP traffic in the Melbourne trace, as it contains 2 million flows which can be handled by our hardware. As we see from Figure 12, all our schemes perform significantly better than the normal scheme when the hash table size is small. Generally, the small hash table size will increase the length of the hash chain, which in turn increases the processing time for each flow state lookup. However, this processing time is reduced if we keep reordering the hash chain to make sure all the active flows are in the front. More importantly, the *insert and move to front* scheme is insensitive to the hash table size as it fully exploits the temporal localities in the traffic. The processing time of all the four schemes converges when a large hash table size is used. This is due to the fact that the average size of the hash chain is reduced to be 1, and there is no difference between the four schemes.

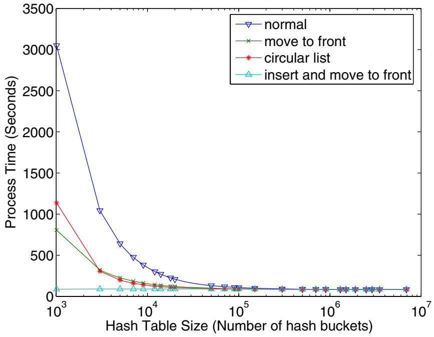


Fig. 12. Processor time used by each scheme for TCP traffic in the Melbourne trace, where no flows are purged from the hash bucket

4.4 Complexity of the Schemes

It is difficult to discuss the complexities of the schemes without knowing the traffic flow statistics and hash function properties. Let us define a simple uniform hash function[2]

$$h : K \longmapsto T$$

where K is the set of keys (the 13 bytes flow identification in our experiment), where T is the set of memory locations in the hash table. Let n be the hash table size and l be the number of keys we want to hash, then the *load factor* can be defined as $\alpha = \frac{l}{n}$. Based on our observation on the Auckland traces and Melbourne traces, we assume the packet sequence within each hash bucket is as follows:
$A_1A_2...A_kB_1B_2...B_kC_1C_2...C_k...A_{(j-1)k+1}A_{(j-1)k+2}...A_{jk}B_{(j-1)k+1}B_{(j-1)k+2}...$ $B_{jk}C_{(j-1)k+1}C_{(j-1)k+2}...C_{jk}(j \gg 1, k \gg 1)$, where A, B, C are the flows that are mapped into the same hash bucket, k is the size of the *burst* and j is the number of bursts in each flow. Then we can summarize the searching complexity of the normal scheme and our schemes in Table 2, where X_i $(i = 0, 1, 2, 3)$ is a variable representing the cost of searching for an element in a hash chain, and

[2] A *Simple Uniform Hash function* assumes that any given element is equally likely to hash into any one of the hash buckets.

Table 2. Comparison between our three schemes and the normal scheme in terms of searching complexity

Normal	Move to front	Circular list	Insert & move to front
$O(1 + X_0\alpha)$	$O(1 + \frac{1}{k}X_1\alpha + \frac{1}{k}Y_1)$	$O(1 + \frac{1}{k}X_2\alpha + \frac{1}{k}Y_2)$	$O(1 + \frac{1}{k}X_3\alpha + \frac{1}{k}Y_3)$

Y_i $(i = 1, 2, 3)$ is a variable representing the cost of reordering the hash chain. Note that $i = 0$ is the *normal* scheme, $i = 1$ is the *move to front* scheme, $i = 2$ is the *circular list* scheme and $i = 3$ is the *insert and move to front* scheme. Based on the analysis of Section 4.1, the variables satisfy $X_0 \approx X_2 > X_1 > X_3$ and $Y_1 = Y_3 > Y_2$. We can see that all our schemes outperform the normal scheme in terms of complexity if the burst length k is sufficiently large.

5 Discussion

From Figure 8 and Figure 10, we can see that our three proposed hash chain management schemes perform better in the Auckland traces than in the Melbourne traces. For example, in the Auckland trace, the circular list scheme reduced memory comparison instructions by 55% compared with the normal scheme; while in the Melbourne trace, it only reduced the memory comparison instructions by 40% compared with the normal scheme.

The major reason for this performance difference is the difference in traffic characteristics. As we analyzed in Section 3, our three proposed schemes perform well if the flows are not highly interleaved, i.e., if the packets of each flow tend to arrive in uninterrupted bursts. The Auckland trace is uni-directional while the Melbourne trace is bi-directional. For the TCP traffic in the bi-directional trace, the incoming flows and the outgoing flows are highly correlated. Hence, the flows in the Melbourne trace are more heavily interleaved than the Auckland trace.

More importantly, the Melbourne trace is collected at the router of the computer science department in the University of Melbourne. Hence, it contains a lot of inter-departmental traffic, and only 50% of the total traffic is TCP traffic. Moreover, nearly 16% of the TCP traffic is SSH traffic. In contrast, the Auckland traces are collected at the router that provides the Internet connection to the University of Auckland. More than 95% of the Auckland trace is TCP traffic, while only 0.25% of the TCP traffic is SSH traffic. Generally, SSH traffic is generated by users who remotely login to a server. The speed of the SSH flows are strongly affected by many human factors, e.g., a person's typing speed or the time spent thinking between typing commands. Moreover, the SSH flows can be active for quite a long time, e.g., researchers may login to servers for several days to run some time-consuming experiments. In summary, the SSH traffic can be described as slow speed flows that last for a long time. Consequently, we are less likely to see flow bursts in the SSH traffic, as slow flows are more likely to be highly interleaved.

From Figure 4 and 8, we can see that the number of memory comparison instructions used in TCP traffic is close to the number of memory comparison instructions used in IP traffic. This is because the majority (over 95%) of traffic in the Auckland traces is TCP traffic. From Figures 6 and 10, we can see that the number of memory comparison instructions used in TCP traffic is less than 20% of the number of memory comparison instructions used in IP traffic. This can be explained by the following two reasons. First, the TCP traffic only accounts for 50% of the total IP traffic in the Melbourne traces. Second, we found a large proportion of UDP scan traffic in the Melbourne traces. The UDP scan traffic generally consists of a large number of short UDP flows, which adds extra overhead to the flow state lookup for the Melbourne traces.

6 Conclusion

In this paper, we have proposed three hash chain management schemes to improve flow state table lookup performance by exploiting the temporal locality in network traffic. Our *insert and move to front* scheme which inserts the new flow and moves the matched flow to the front of the hash chain, performs the best. It is followed by our *circular list* scheme and *move to front* scheme. The *insert and move to front* scheme has the best performance and is insensitive to the hash table size as it constantly reorders the hash chain to accurately reflect the locality of the incoming traffic. The circular list approach only preserves the locality for the matched node, where the localities of rest of the flows in the bucket are lost. By evaluating our schemes using real life data traces, we see that all our three schemes perform better than the normal scheme. We also discussed several factors that affect the flow state table lookup performance, such as the type of traffic. The conclusions of our research are now being incorporated by our industry partner (Intelliguard Pty Ltd) into their network security products.

Acknowledgment

We would like to thank the ARC Special Research Center for Ultra-Broadband Information Networks (CUBIN) for providing the test environment, and thank the Waikato Applied Network Dynamics Research Group for making available their data traces. We also wish to thank Mr. Darren Reid and Mr. Peter Yandell for their useful suggestions and comments. This work was supported by the Australia Research Council.

References

1. Waikato Applied Network Dynamics Research Group, The University of Waikato.
2. http://tcpreplay.sourceforge.net/.
3. Packet Filter. http://www.benzedrine.cx/pf.html.

4. S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. RFC 2401, the Internet Engineering Task Force (IETF), December 1998.
5. P. Gupta and N. McKeown. Algorithms for packet classification. *IEEE Network*, March 2001.
6. Darren Reed. IP Packet Filter. http://coombs.anu.edu.au/˜avalon/.
7. J. Zobel, S. Heinz, and H. E. Williams. In-memory hash tables for accumulating text vocabularies. *Information Processing Letters*, 80(6):271–277, 2001.

The Interaction of Forward Error Correction and Active Queue Management

Tigist Alemu, Yvan Calas, and Alain Jean-Marie

LIRMM UMR 5506 CNRS and University of Montpellier II,
161, Rue Ada, 34392 Montpellier Cedex 5, France
Tel: (33) 4 67 41 86 02 Fax: (33) 4 67 41 85 00
{tigist, calas, ajm}@lirmm.fr

Abstract. This paper studies the interaction of a forward error correction (FEC) code with queue management schemes like Drop Tail (DT) and RED. Since RED spreads randomly packet drops, it reduces consecutive losses. This property makes RED compatible *a priori* with the use of FEC at the packet level. We show, through simulations, that FEC combined with RED may indeed be more efficient than FEC combined with DT. This however depends on several parameters like the burstiness of the background traffic, the FEC block size and the amount of redundancy in a FEC block. We conclude generally that using FEC is more efficient with RED than with DT when the loss rate is small, a relatively important amount of redundancy and at most a moderate FEC block size is used. We complement these observations with a simple model, which is able to capture the tradeoff between the locality and the frequency of losses.

Keywords: RED, FEC, queue management.

1 Introduction

The Internet traffic suffers from heavy losses due to network congestion caused by the limited capacity of queue in the routers. There exist two end-to-end error control techniques to repair these losses: ARQ (Automatic Repeat reQuest) which consist in retransmitting dropped packets upon the destination's request, and FEC (Forward Error Correction) which consists in sending redundant packets to the destination, allowing it to repair losses without requiring packet retransmission. Because of retransmissions, ARQ is not appropriate for real-time applications. This is why FEC is increasingly used in such applications, typically on top of the UDP transport protocol. FEC is also used in bulk data transfer applications such as Digital Fountain [1]. Several drawbacks are attached to the use of FEC. First, FEC cannot recover all lost packets. In addition, the transmission of redundant packets increases the overall network load. Finally, the effectiveness of FEC is known to depend on the way packet drops are distributed in the data stream. FEC is more efficient when packets losses are independent, and much less when they occur in groups [2].

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1146–1155, 2005.

In conjunction to end-to-end error control techniques, there exist queue management schemes operating inside routers that control network congestion. The “queue management” scheme traditionally used in the current Internet is Drop Tail (DT), which consists in discarding arriving packets when the buffer of the router overflows. *Active* queue management schemes, in particular the Random Early Detection (RED) scheme [3, 4], have been recommended recently by the IETF as an alternative, aimed at eliminating deficiencies of Drop Tail. The RED scheme basically discards packets earlier so that incipient stages of congestion can be detected.

The aim of this paper is to study the interaction of FEC with RED (RED/FEC) and to compare the obtained results with those obtained from the combination of FEC with Drop Tail (DT/FEC). This study has never been conducted to our knowledge. Indeed, FEC has always been studied in presence of the Drop Tail queue management. We believe that as compared to Drop Tail, RED may give performance improvement for the UDP sources implementing FEC since it spreads randomly packet drops between flows, reducing consecutive losses for a given flow, thereby making losses “more independent”.

In Section 2, we briefly present the principles of the FEC coding scheme and of queue management algorithms. In Section 3, we describe the topology of the system and the performance metrics considered in this paper. The performance measures obtained by simulation are detailed and analyzed in Section 4. Section 5 presents a simple model with which we explain the tradeoff responsible for the fact that sometimes RED/FEC is more efficient, sometimes DT/FEC. Section 6 presents conclusions and perspectives.

2 FEC and Queue Management

We first recall some properties of codes used for Forward Error Correction. Given k data packets bearing the relevant information, the encoder (based for instance on a Reed-Solomon Erasure code [5]) generates h redundant packets useful for the recovery of the lost data packets. The concatenation of the k data packets and the h redundancy packets is called a *FEC block* of size $n = k + h$. If the total number of lost (data and redundant) packets is at most h, the decoder at the destination can retrieve successfully all lost packets. As a result all the relevant information is saved. Otherwise, if the total loss exceeds h packets, it is impossible to recover the lost packets.

The queue management schemes studied here are Drop Tail, the principle of which is straightforward, and RED. With RED, the router maintains an estimate of the average queue length, using an exponential moving average. Based on this value, it accepts or rejects incoming packets with a certain probability. The rejection probability function is a parameter of the mechanism. We have used in the following experiments the default values for RED parameters [3, 6].

3 Experimental Setup

3.1 Network Topology

The network setup is depicted in Figure 1. The traffic generated by nodes S_0 to S_N is multiplexed on a 10 $Mbps$ bottleneck link between nodes R_1 and R_2 with a propagation delay of $30ms$. The bottleneck link is provided with a Drop Tail or a RED queue of limited capacity of 35 packets. The other links located between nodes $S_0, \ldots, S_N$ and node R_1 have a capacity of $100Mbps$. These links have different propagation delays uniformly distributed from $20ms$ to $100ms$.

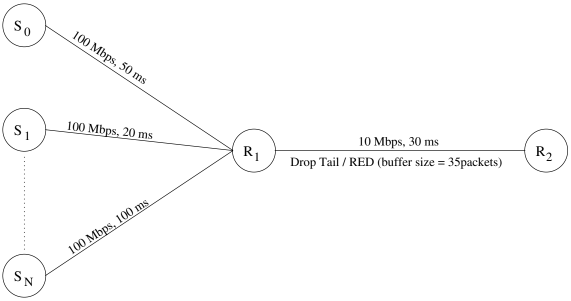


Fig. 1. Topology of the system

We have studied a traffic mix of TCP and UDP flows, with UDP representing a minority of the traffic. Our purpose in doing so is that we wish to study the behavior of the FEC technique under bursty conditions. A background traffic generated by TCP sources is appropriate in that respect since TCP traffic is usually quite bursty. Alternately, a simulation with variable-rate UDP sources could have been used. In the future of networks, it may be that flows of real-time applications (such as our UDP sources) will be separated from elastic TCP traffic, in which case the network conditions encountered by the real-time traffic could be quite different. But in the current Internet, where service differentiation is not yet widespread, the experimental setup we have chosen represents an average situation. The study of these conditions can actually help deciding if deploying flow differentiation at the node level can be dispensed with.

A Poisson process is used for the generation of the foreground UDP traffic at node S_0. Node S_1 to S_N generate background long-lived FTP traffics using TCP/Sack1 agents. The offered load of the UDP traffic in the absence of redundancy is set to $\rho_1 = 500kb/s$. Without redundancy the load generated by the UDP traffic represents 5% of the bandwidth of the bottleneck link. For the generation of redundant packets, we increase the throughput ρ_1 of the UDP/FEC flow by a factor of $1 + h/k$, in order to take into account the addition of redundancy while keeping constant the rate of information generated by the source. The resulting new load ρ_{FEC} is equal to $\rho_{FEC} = \rho_1(1 + h/k)$. Under `ns-2` [7], the standard way to generate a Poisson process (approximately) is to use the exponential on/off source. Bursts of packets of size one are obtained with a very large bit rate at the source. The space between packets is controlled with the variable `idle_time_`. In order to take into account the increase of the load due to redundancy, this value is computed as follows:

$$\texttt{idle\_time\_} = \frac{\text{UDP packet size}}{500 \times (1 + h/k)} ,$$

where the packet size for the UDP traffic is set to 573 bytes following the recommendation of [8]. TCP packet sizes are set to 1200 bytes and the maximum TCP window size is set to 20 packets, that is, 24kBytes. Since the delay × bottleneck bandwidth product is 300 kb or 37.5kBytes, this means that a single source cannot saturate the link. Actually, given the maximum RTT of 260ms, the maximum offered throughput of one source is about 100kBytes/s. The superposition of about 12 sources should saturate the 10Mbps link. In addition, assuming a full window, the typical sending pattern of a source should be 20 packets simultaneously each RTT, thus realizing a bursty arrival pattern at the bottleneck, as intended.

Every simulation is run for 100 simulated seconds and statistics are collected every $10ms$ from the queue located between node R_1 and R_2. The results presented below are averages over 50 independent simulations.

3.2 Performance Metrics

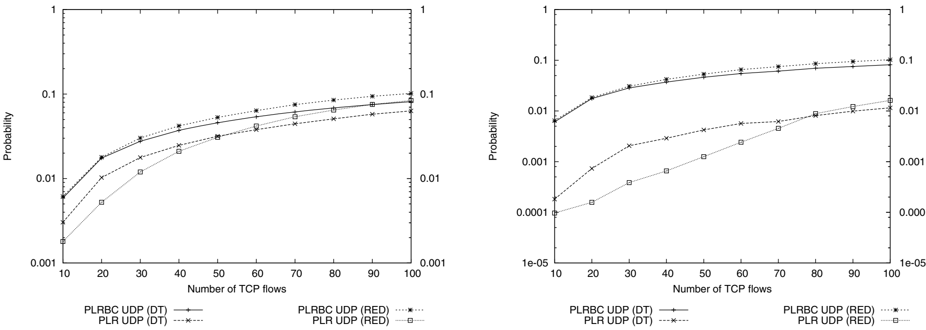
The metrics used for a specific flow (that is for the FEC flow) are:

- The *packet loss rate before correction* (*PLRBC*) that is the ratio of the average number of lost packets in a FEC block before correction to the size of the FEC block.
- The *packet loss rate after correction* (*PLR*) that is the ratio of the average number of lost packets in a FEC block after correction to the size of the FEC block.
- The *loss run length* [9] that is the number of packets of a particular flow that are lost consecutively. This is a random variable which gives an insight into the packet loss process. Studying this metric allows to investigate which queue management is more efficient when used with FEC.

4 Performance Measures

4.1 Influence of the Number of TCP Flows

Figure 2 shows the evolution of the packet loss rate before correction (PLRBC) and after correction (PLR) for the UDP source as a function of the number of TCP flows and the number of redundancy h. As observed in [10] and as shown also in Figure 2, when the amount of redundancy is increased, the PLR of DT/FEC decreases for this network configuration, *i.e.* for a configuration where the number of sources implementing FEC is small. This observation is maintained for the PLR of RED/FEC in this case.



(a) $k = 16, h = 1$. (b) $k = 16, h = 4$.

Fig. 2. Packet loss rate before (PLRBC) and after (PLR) correction by FEC

As expected and as shown by [11], we observe that the PLRBC for RED is greater than the PLRBC for Drop Tail since RED starts dropping packets earlier without reaching the buffer capacity of the queue. Nevertheless, after the correction of lost packets, RED/FEC out-performs DT/FEC and gives a lower PLR for a small number of TCP flows. For a larger number of flows, the situation is reversed: RED yields a worst performance. Indeed, for one packet of redundancy ($h = 1$) and $k = 16$ data packets, *i.e.* for an addition of 6% of load, Drop Tail can divide the PLRBC by about 1.9 for 10 TCP flows. RED does better by dividing the PLRBC by about 3.4. In the case of a 25% load increase, for $h = 4$ and $k = 16$, Drop Tail can divide the PLRBC by about 33.4 for 10 TCP flows. In this case, the correction rate of RED is more significant since it is able to divide the PLRBC by 66.5 for 10 TCP flows and by 79.6 for 30 TCP flows.

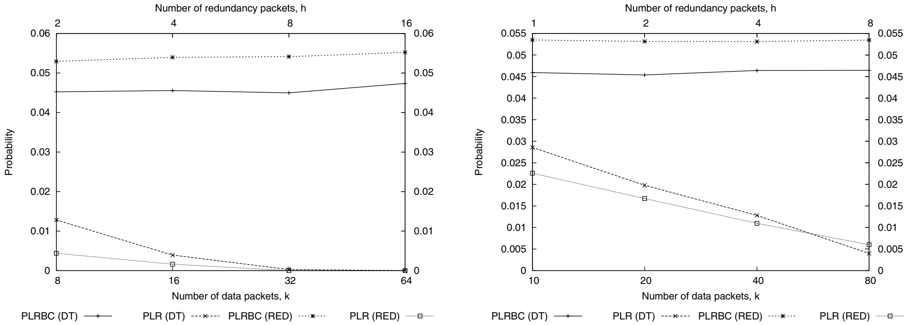
These results show that the number of TCP flows under which RED experiences an improvement on the PLR as compared to Drop Tail depends on the amount of redundancy. Hence, this threshold number increases as the amount of redundancy increases. For instance, this number is: 45 flows for $h = 1$ and 80 flows for $h = 4$. But the increase becomes slower as h grows. We have therefore shown that even if RED increases the UDP packet loss rate (which is in accordance with previous studies [11, 12]), it becomes possible to reduce its PLR by using FEC and to obtain a PLR less than the PLR for Drop Tail under certain conditions (precisely for a certain number of TCP flows and a certain amount of redundancy).

4.2 Loss Run Length

We have also studied the distribution of the loss run length, for both queue management mechanisms and in function of the cross traffic. The results fully reported in [13] showed that for $k = 16$ and $h = 1$, under the RED scheme, the probability to have a loss run length of size 1 packet is much larger than under Drop Tail (about 90% against 60%, respectively). This holds whatever the number of flows: the distribution does not appear to depend much on the cross traffic. This last observation shows that the situation is not as simple as initially thought. Our starting assumption was: if RED shows a smaller loss run length, then FEC will perform better with RED than with Drop Tail concerning the capacity of repairing lost packets. This turns out not to be valid for a large cross traffic, although the loss run length of RED is small throughout the range of experiments. We develop in Section 5 a model which shows that there is actually an efficiency tradeoff between the *size of bursts* of lost packets, and their *frequency*.

4.3 Influence of Redundancy and FEC Block Size

In this experiment, we made the size of the FEC block vary, while maintaining the rate h/k constant, in order to obtain the same load increase for the UDP flow and therefore maintain the same overall network load. This way we can directly observe the influence of the FEC block size without the interference of the network load. We have also studied in [13] the case of a variable UDP load.



(a) Block size variation with constant rate $\frac{h}{k} = 0.25$ and with 50 TCP sources.

(b) Block size variation with constant rate $\frac{h}{k} = 0.1$ and with 50 TCP sources.

Fig. 3. Influence of the block size

The load of the system being constant in this case, the PLRBC is fixed for both RED and Drop Tail as illustrated by Figure 3. However, the PLR decreases when k (and therefore h) increases for both queue management schemes. This means that for this configuration of the system, it is interesting to increase the FEC block size. We also notice in Figure 3 that RED/FEC appears to be more advantageous than DT/FEC concerning the PLR under certain conditions depending on the number of TCP flows, the FEC block size and the amount of redundancy. This is the case in Figure 3(a), even though the PLRBC of RED/FEC is larger than the one obtained for DT/FEC. Indeed, RED queue management is able to give a slight performance improvement.

To summarize, all these results suggest that in case of a constant UDP offered load, the larger the block size, the better the correction rate is for both RED and Drop Tail. It is more interesting to use FEC with RED instead of Drop Tail in case of small number of flows, moderate FEC block size, and a relatively important amount of redundancy. Indeed, our model developed in Section 5 confirmed that RED should be more efficient if the redundancy ratio h/k is sufficiently large. Additional experiments conducted in [13] have shown that when the cross traffic is large, RED/FEC loses its advantage over DT/FEC whatever the FEC block size and the amount of redundancy.

5 A Model for FEC and Its Application

We develop in this section a simplified model which is able to explain most of the phenomena observed above. This model concentrates on the sequence of packets and among them, the lost packets.

The sequence numbers $m_1, m_2, \ldots$ of the lost packets can be viewed as the instants of an "arrival" process. Call a *block* of losses a set of packets lost consecutively. If the size of blocks of losses is small compared with the time between two of these blocks, then an approximation of the situation is a *batch* arrival process, where several packets are lost simultaneously.

The most tractable of such processes is the one where grouped losses occur according to a Poisson process of rate λ. Assume that the size of the m-th group is a random variable $A_m > 0$. The sequence $\{A_m\}_m$ is assumed to be i.i.d., and we shall use the notation A for the generic random variable. The resulting process is a compound Poisson process, and the distribution of N_T, the total number of lost packets in the interval $[0, T]$, can be computed as:

$$P(N_T \leq h) \;=\; \sum_{m=0}^{\infty} \frac{(\lambda T)^m}{m!} e^{-\lambda T} P(A_1 + \ldots + A_m \leq h) \; . \tag{1}$$

The expected number of losses per unit time, which is to be interpreted as a loss rate of packets, is $p = \lambda E(A)$.

Consider now two situations where the distribution of the batch size differs, but where the average loss rate p is the same. Quantities referring to situation i will be superscripted with "(i)". Coming back to the focus of this paper, we consider that the first situation is RED/FEC and that the second is DT/FEC. We choose a simple distribution for the batch size: $A = 1$ with probability β and $A = 2$ with probability $1 - \beta$. The frequency of blocks of losses of size 1 is therefore β. According to the measurements of Section 4.2, the situations of RED and DT correspond approximately to values $\beta^{(1)} = 9/10$, and $\beta^{(2)} = 6/10$, respectively.

Assume a certain fixed T and some integer h, interpreted as the length of some FEC block, and the quantity of redundancy it contains, respectively. We are interested in the difference between the probabilities of repairing this FEC block in both situations, when the packet loss rate is p. It can be expressed as:

$$\mathit{\Delta}_h(x) \;=\; P(N_T^{(1)} \leq h) \;-\; P(N_T^{(2)} \leq h)\;, \tag{2}$$

where both probabilities are obtained using (1) with $\lambda = p/m^{(i)}$ (denoting $m^{(i)} = E(A^{(i)})$), since we have assumed the same loss rate p in both situations. The difference is a function of $x = pT$, the average number of lost packets in the interval $[0, T]$.

Plotting the functions $\mathit{\Delta}_h(x)$, we find that for each h there exists a threshold value x_h such that $\mathit{\Delta}_h(x) \geq 0$ when and only when $x \leq x_h$. This difference is therefore positive if $x = pT$ is small enough (RED/FEC has a better performance), negative if x is large (DT/FEC has a better performance).

The explanation for the shape of the functions $\mathit{\Delta}_h$ is found analyzing Equation (1). Indeed, when $x = pT$ is small, the difference is dominated by polynomial terms in x, with coefficients $P(A_1^{(i)} + \ldots + A_m^{(i)} \leq h)$. Because $A^{(1)}$ is stochastically larger than $A^{(2)}$, this implies that $P(N_T^{(1)} \leq h)$ is larger. Here, the fact that blocks are smaller is important. On the other hand, if x is large, the terms in (2) are dominated by $e^{-x/m^{(i)}}$. Since $m^{(1)}$ is smaller, $P(N_T^{(1)} \leq h)$ decreases faster and gets smaller than $P(N_T^{(2)} \leq h)$. Here, the fact that blocks of losses are less frequent is more important.

Computing numerically these threshold values further reveals that when h is large enough, $x_h \simeq h + C$ where C is some positive constant value. Using this empirical finding, and since the size of the block T is $k + h$, we have: RED/FEC is better if

$$\begin{aligned} p(k+h) \;\leq\; h + C \;\Longleftrightarrow\; k \;&\leq\; \frac{1-p}{p}\,h + \frac{C}{p} \\ \Longleftrightarrow\; \frac{h}{k} \;&\geq\; \frac{p}{1-p} - \frac{C}{1-p}\frac{1}{k}\;. \end{aligned}$$

This model allows therefore to predict that: a) for a given quantity of redundancy h, RED/FEC is more efficient only for a block size k small enough; b) RED/FEC is more efficient only if the redundancy ratio h/k is large enough; c) the larger the loss rate p, the smaller the block size k of RED/FEC can be. All these qualitative predictions are confirmed by simulation experiments reported in Section 4.3 or in [13]. For instance, we have in Figure 3 situations where RED/FEC as the advantage for block sizes small enough. In other cases reported in [13], h/k is too small to compensate the loss rate.

To conclude, we observe that this preliminary model is too crude to produce accurate quantitative predictions, at least for the network simulated in Section 4. In addition, in these experiments, the loss rate (or PLRBC) p is not the same for both situations. We believe however that this first simple model reproduces adequately the principal features of the problem. Moreover, it can be checked that the principal conclusions hold when the loss rate of RED is increased (the range of parameters where RED/FEC performs better is then reduced). We have also found that an analysis based on the classical Gilbert model also exhibits the performance inversion which we have discussed here.

6 Conclusions and Perspectives

In this paper, we have studied the effect of a forward error correction (FEC) code on queue management schemes like Drop Tail and RED. It should be noted that to the best of our knowledge, no study has been conducted so far concerning FEC combined with a RED-like active queue management scheme.

It has been shown in literature that RED losses are spread as compared to Drop Tail. For this reason, one can assume that FEC would be more efficient combined with RED than with Drop Tail. But our results have also shown that RED/FEC does not always perform better than DT/FEC. This turns out to depend on certain parameters, in particular on the number of TCP flows that constitute the background traffic, the FEC block size and the amount of redundancy in a FEC block.

Our results suggest that RED/FEC performs better than DT/FEC when the loss rate is not too large, when there is a relatively important amount of redundancy in the FEC block, and a moderate FEC block size is used.

From the point of view of the queue manager, our results suggest the possibility to swap between RED and Drop Tail depending on the scenario parameters. For instance, based on the estimated number of TCP flows or the observed PLRBC, and knowing that FEC is implemented in the UDP flows, the queue manager can take a decision. If the PLRBC is high, it is more advantageous to use Drop Tail. Otherwise, when the PLRBC is low, then RED scheme can be used by following the guidelines described just above.

Our next step will be to refine the model of Section 5 in order to investigate further the phenomenon we have observed, in which the correction capacity of FEC is worst when coupled with RED, despite the fact that losses are almost always isolated. First, it is necessary to prove the properties on the threshold values x_h which we have only empirically described. Next, we will investigate whether these properties hold with more general batch size distributions, and can be exploited to obtain decision rules concerning the use of FEC.

References

1. Byers, J., Luby, M., Mitzenmacher, M., Rege, A.: A digital fountain approach to reliable distribution of bulk data. In: Proc. ACM SIGCOMM, Vancouver, Canada (1998) 56–67
2. Cidon, I., Khamisy, A., Sidi, M.: Analysis of packet loss processes in high-speed networks. IEEE Transactions on Information Theory **39** (1993) 98–108
3. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking **1** (1993) 397–413
4. Braden, B., et al.: Recommendations on queue management and congestion avoidance in the Internet (RFC 2309). IETF. (1998)
5. McAuley, A.: Reliable broadband communications using a burst erasure correcting code. In: Proc. ACM SIGCOMM'90, Philadelphia, PA, USA (1990) 297–306
6. Floyd, S.: Discussions on setting RED parameters (1997) http://www.aciri.org/floyd/red.html.

7. NS: Simulator homepage (2004) `http://www.isi.edu/nsnam/ns`.
8. Brandauer, C., Iannaccone, G., Diot, C., Ziegler, T., Fdida, S., May, M.: Comparison of tail drop and active queue management performance for bulk-data and web-like Internet traffic. In: Proc. ISCC'01, Hammamet, Tunisia (2001)
9. Sanneck, H., Carle, G.: A framework model for packet loss metrics based on runlengths. In: Proc. Multimedia Computing and Networking Conference (MMCM), San Jose, CA, USA (2000) 177–187
10. Biersack, E.: A simulation study of forward error correction in ATM networks. Computer Communication Review **22** (1992) 36–47
11. Bonald, T., May, M.: Drop behavior of RED for bursty and smooth traffic. In: Proc. IEEE/IFIP IWQoS'99. (1999)
12. May, M., Bonald, T., Bolot, J.: Analytic evaluation of RED performance. In: Proc. INFOCOM'00. (2000)
13. Alemu, T.: Performance evaluation of quality of service mechanisms in the Internet. PhD thesis, University of Montpellier II (2004)

ELIP: Embedded Location Information Protocol

Farid Benbadis, Marcelo Dias de Amorim, and Serge Fdida

Laboratoire LIP6/CNRS,
Université Pierre et Marie Curie,
8, rue du Capitaine Scott – 75015 – Paris – France
{benbadis, amorim, sf}@rp.lip6.fr

Abstract. It has been recently shown that mobility in ad hoc networks can be an advantage instead of an inconvenience. Nevertheless, one class of mobile elements has been neglected up-to-date: *data packets*. In this paper, we propose to take advantage of the inherent mobility of data packets to disseminate location information throughout the network. We focus on the age-and-position based (APB) routing case. Knowing its own geographic or virtual coordinates is not enough since a source needs to discover the position of the destination before establishing a communication. This is the role of a location service, which depends, in turn, on an efficient location distribution/publishing system. Our proposal, Embedded Location Information Protocol (ELIP), allows nodes to piggyback their coordinates in existing data packets in order to efficiently disseminate their positions in the network. Contrary to traditional approaches that depend on encounters between nodes, ELIP converges much faster and does not require permanent node mobility.

1 Introduction

In large scale mobile ad hoc networks, position-based routing has proven to be efficient because of its simple forwarding policies. Indeed, nodes make elementary forwarding decisions based solely on the coordinates of their direct neighbors and of the destination. This avoids the need for topology knowledge beyond one-hop. Since there is no need for maintaining explicit routes, this type of routing algorithm is scalable and robust to mobility.

Position-based routing algorithms are composed of three main steps. The first one is *positioning*, where each node determines its coordinates using an absolute positioning system like GPS (Global Positioning System) or a GPS-free relative positioning algorithm [1, 2, 3]. The second step is the *location service*, used by a source to obtain the destination's coordinates. It is important to note here that the location service is composed of two distinct operations (cf. Section 2): dissemination of location information and lookup. The third step is *forwarding*, where next-hop decisions are based on the destination's position and on the position of the forwarding node's neighbors. Examples are the Compass routing [4], Restricted Directional Area, used in [5], or Greedy Perimeter Stateless Routing (GPSR) [6]. Refer to [7] for a comprehensive survey on such algorithms.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1156–1167, 2005.

In this paper, we focus on the *dissemination* model associated with the location service. The dissemination model dictates the degree of location information replication throughout the network. This replication may range from null, meaning that a node is the only one to know its position, to full, where all nodes know the entire topology. Both cases have pros and cons. On the one side, zero replication does not require updates of information that are never used, but results in a lookup procedure that generates a great amount of control traffic overhead. On the other side, total replication results in zero overhead for lookups but high overhead for maintenance, especially in networks of mobile nodes.

A more recent way of estimating locations in ad hoc networks without incurring much traffic overhead is to use age-and-position based (APB) algorithms. In such algorithms, every node maintains a local database where it records the identifier and location of other nodes in the topology. Each location is associated with an *age*, which gives the time elapsed since the last time the location information has been updated. The local database is consulted to obtain approximate coordinates of the destination's current position. In such an approach, a node sends packets to the destination's position it knows. These packets are rerouted by nodes that have fresher location information until they are received by the destination [8]. Contrary to traditional approaches where the data transmission phase comes after the location phase, in APB methods the destination's position discovery is achieved during packet forwarding. It is clear that the lower the age, the better the estimation of the node's location. The problem in APBs is then to find an efficient way of distributing good estimations of node positions in the network.

In [8], Grossglauser and Vetterli propose to use encounters as a way of disseminating location information. In such an approach, nodes update their local databases each time they are directly connected to other nodes. The advantage of using encounters is that it results in near-zero location dissemination overhead. Nevertheless, as we will see in Sections 2 and 4, it results in high overhead in the lookup phase. Furthermore, the efficiency of this approach is closely related to the mobility model of the nodes.

In this paper, we do not propose a novel APB routing protocol, but an efficient dissemination mechanism of nodes' coordinates that can be used in any APB protocol. This algorithm is *Embedded Location Information Protocol* (ELIP). It uses *existing* data packets to disseminate location information. ELIP relies on the basic assumption that packets are much more mobile than nodes. ELIP piggybacks nodes coordinates with existing data packets, incurring little traffic overhead to better disseminate topology knowledge than encounter methods. We will show that, although ELIP incurs little overhead for disseminating location information, it largely reduces the global overhead (dissemination+lookup) when compared to the encounter approach.

Our algorithm is efficient for these three reasons: (1) a node communicates with several other nodes. Thus, it is likely that it sends packets containing its coordinates in different directions, which results in packets traveling across several other nodes in the topology; (2) nodes move and forward packets in different regions of the topology, which leads to a wider dissemination pattern; (3) the overhead generated by the headers insertion is negligible when compared to the reduction of overhead signaling messages during the lookup phase.

Our results show that using ELIP, instead of encounters, leads to much lower average ages of location information in the network. The consequence is that the lookup phase generates lower discovery traffic overhead and delay. Furthermore, we observe that the resulting path lengths are about 30% shorter with ELIP. We also show through a number of simulations that the global overhead in ELIP is upper bounded by the encounter-based approach.

The remainder of this paper is organized as follows. In Section 2, we present the context where the ELIP algorithm can be used. Section 3 details the main components of ELIP and how nodes use existing data packets to disseminate location information. In Section 4, we evaluate the performance of our proposal with different topology densities and mobility models, and compare it with the encounter-based approach. Finally, Section 5 concludes the paper and provides some ideas for future work.

2 Location = Dissemination + Lookup

A location service is composed of two main components: dissemination and lookup. The dissemination model refers to the ability of the location service to distribute location information throughout the network. ELIP falls into this category. The lookup phase consists in obtaining the location information of a destination from a node that has been provided with this information during the dissemination phase. In the following we will present different ways of implementing a dissemination model as well as the lookup algorithm that we will use to evaluate ELIP.

2.1 Disseminating Location Information

As stated above, the efficiency of the location service depends on an adequate system to disseminate location information in the network. This task can be performed in different ways according to how many nodes play the role of a location server. In *all-one* approaches, every node knows only its own position. This method requires that the source floods a route request in the network until the destination responds with its coordinates. This is clearly not scalable because of the high traffic control overhead generated. Examples of protocols that use such an approach are DSR [9] and AODV [10]. *Some-some* methods distribute the whole topology information among a subset of the nodes. When a node looks for a destination, it sends a route request to one of the servers. Although simple, this approach is considered unfair in the ad hoc concept because some nodes have more responsibilities than others [7]. In the *all-some* algorithms, every node in the topology plays the role of a rendezvous point. This category includes DHT-based location services, where a node n stores its location information in a rendezvous node r depending on n's identifier [11, 12].[1] The problem in such a system is that, in order to keep an accurate location system, the location information must be updated every time a node moves. Depending on the dynamic nature of the topology, this may lead to high signaling overhead. Finally, there is the *all-all* approach, where each node always knows the positions of all the other nodes in the topology in a proactive fashion [13]. While discovering a node's position is fast and does not generate any traffic overhead, such a solution leads to high traffic overhead in order to keep databases constantly updated.

2.2 Age- and Position-Based (APB) Routing

In APB routing protocols, we essentially use the age of location information to compute routes from source to destination. Because nodes move, their positions change with time. Location information of node m stored in n must be then frequently updated. The older the information, the worse the location estimation.

The Last Encounter Routing (LER) has been proposed by Grossglauser and Vetterli as an implementation of the APB algorithm [8]. In LER, nodes do not exchange any explicit location information. The only available information a node has is the history of its encounters with other nodes and the ages of these encounters. They assume w.l.g. that two nodes have encountered each other if they have been directly connected in the past. In their proposal, the authors implement LER by using Last Encounters (LE) as the dissemination algorithm and Exponential Age SEarch (EASE) as the lookup method.

Our proposal, ELIP, is an alternative to LE. Thus, in order to fairly compare these two approaches, we use EASE as the lookup algorithm. We briefly describe EASE in the following. For further details, please refer to [8].

For two arbitrary nodes i and j, we note $\tau_{i,j}$ the time elapsed since the last time i and j were directly connected, with the convention that $\tau_{i,j} = \infty$ if i and j have never met, and $\tau_{i,j} = 0$ if they are currently connected. We also note $l_{i,j}$ the position where i has met j for the last time.

Consider a node s that wants to communicate with d. If $\tau_{s,d} \neq \infty$, then s sends the message to $l_{s,d}$, *i.e.* the d's coordinates stored by s in its position table.[2] The node the geographically closest to $l_{s,d}$, called *anchor node*, will be responsible for determining the following coordinates the message must be sent to (*i.e.* the following anchor node). The idea is to make messages jump between anchor nodes with decreasing ages until the message is received by the destination.

Let $\mathbf{A} = \{a_1, a_2, a_3, \ldots, \}$ be the set of anchor nodes for a message traveling from s to d. Let also $\tau^{\ominus}_{a_i}$ be the age indicated in the message received by anchor node a_i and $\tau^{\oplus}_{a_i}$ be the age of the new location information that anchor node a_i must compute. Clearly, we must guarantee that $\tau^{\oplus}_{a_i} < \tau^{\ominus}_{a_i}$.

The first test node a_i makes is to look into its own position table to verify if $\tau_{a_i,d} < \tau^{\ominus}_{a_i}$. If so, then a_i does $\tau^{\oplus}_{a_i} := \tau_{a_i,d}$ and sends the packet to $l_{a_i,d}$.

[1] DHT = Distributed Hash Table.

[2] We assume that the message is routed from l_s to $l_{s,d}$ via a geographic forwarding protocol. We do not focus on this point in this paper.

Otherwise, it locally floods a d's *position request* with some TTL (time-to-live). Node a_i receives a response only if at least one of the nodes within this scope has encountered d more recently than $\tau_{a_i}^{\ominus}$. If a_i does not receive any response, it increases the TTL and floods a second request. This procedure is repeated until node n_i, a neighbor of a_i, responds with $l_{n_i,d}$, *i.e.* the location of its last encounter with d, and $\tau_{n_i,d}$. Node a_i does $\tau_{a_i}^{\oplus} := \tau_{n_i,d}$ and forwards the message to $l_{n_i,d}$, which will be received by the next anchor node a_{i+1}. The same search procedure is performed by a_{i+1}, and so on until the destination d is reached. It has been shown in [8] that the destination is reached after a number of steps of exponentially decreasing distances for a network with random node mobility.

3 Embedded Location Information Protocol: Algorithm Details

Recall that the ELIP algorithm is designed to be integrated in an APB routing protocol for ad hoc environments where nodes are supposed to know their geographic (or virtual) coordinates. The goal of ELIP is to increase the dissemination degree of node location information, making it easier for a source to locate a destination. The originality of ELIP is its ability to widely disseminate location information and reduce the global overhead by piggybacking position and age information of mobile nodes with existing data packets without creating any new signaling packets.

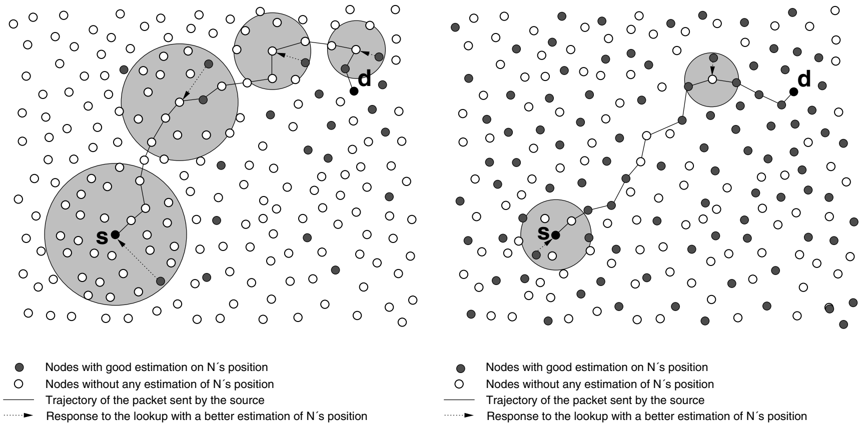
3.1 Node Mobility × Packet Mobility

LER, contrary to traditional routing algorithms that have problems to deal with node mobility, takes advantage from it. The problem of LER is that its performance depends on the mobility pattern of the nodes. If nodes present low mobility or limited occupancy area, they are unable to encounter other nodes, but only a subset of the nodes in the same physical scope.

In ELIP, location dissemination is implemented by encounters between nodes and by including location information in data packets. Because a node communicates with several other nodes, data packets cover a larger area through different routes. All nodes in these routes are then able to update their routing tables even if nodes do not move at all.

Fig. 1 shows an example of the functioning of EASE in two different scenarios. In Fig. 1(a), the distribution of location information about the destination is low. Note that anchor nodes perform searches in relatively large zones, which incurs high signaling traffic overhead. In the second scenario, location information about the destination is well disseminated in the network. This clearly reduces the overhead generated during the lookup phase.

The good performance shown in the example of Fig. 1(b) can be achieved if nodes use ELIP because updated location information is carried by packets, instead of nodes. This is however difficult to obtain when using LE because nodes must permanently travel the entire topology in order to maintain a good


s
d
● Nodes with good estimation on N´s position
○ Nodes without any estimation of N´s position
Trajectory of the packet sent by the source
Response to the lookup with a better estimation of N´s position
s
d
● Nodes with good estimation on N´s position
○ Nodes without any estimation of N´s position
Trajectory of the packet sent by the source
Response to the lookup with a better estimation of N´s position

(a) Low dissemination

(b) High dissemination

Fig. 1. Impact of the dissemination model on the lookup overhead

distribution of location information throughout the network. The gains obtained by ELIP include: smaller latency for location discovery, limited overhead, and shorter end-to-end paths.

3.2 Algorithm

In ELIP, each node in the path between the source and the destination can perform two operations: read and write.

- Read: A node participating in the forwarding procedure consults the ELIP field in the packets and updates the corresponding entry in its position table.
- Write: Any node participating in the forwarding procedure is a candidate node to include its location coordinates in packets. This depends on the insertion model (cf. section 3.2.4) adopted by the nodes.

Let $l_i = (x_i, y_i)$ be the geographic coordinates of node i and $\mathbf{P}_i$ be the position table that i uses to store positions and ages about other nodes (we will explain in details in Section 3.2.3 how i obtains these positions). An example of $\mathbf{P}_i$ is shown in Fig. 2. In this position table, the information node i has about j is $A_{i,j} = [\mathtt{ID}_j, \hat{l}_{i,j}, \tau_{i,j}]$, where $\hat{l}_{i,j}$ is a local approximation of l_j and $\tau_{i,j}$ is the time elapsed since the last time i updated this information (*i.e.*, the age of $\hat{l}_{i,j}$).

Existing data packets are encapsulated in ELIP packets with the structure shown in Fig. 3. In this packet, d is the destination node, l_d is an estimation of d's position, τ_d is the age of this estimation, and $\varGamma$ is the current location of some node in the path traversed by the packet. The node that fills $\varGamma$ depends on the insertion criteria presented in Section 3.2.4. Fields l_d and τ_d are required by the lookup algorithm, which is common to both ELIP and LE. The only overhead introduced by ELIP is the $\varGamma$ field, which contains the ID and the location of the

Node ID: i

Dest. ID	$\hat{l}_{i,\cdot}$	$\tau_{i,\cdot}$
1	$(\hat{x}_{i,1}, \hat{y}_{i,1})$	$\tau_{i,1}$
2	$(?, ?)$	∞
$\vdots$	$\vdots$	$\vdots$
i	(x_i, y_i)	0
$\vdots$	$\vdots$	$\vdots$
j	$(\hat{x}_{i,j}, \hat{y}_{i,j})$	$\tau_{i,j}$
$\vdots$	$\vdots$	$\vdots$
N	$(?, ?)$	∞

Fig. 2. Example of a position table. For the sake of clarity, "$(?, ?)$" and "∞" indicate that node i has no estimation of the corresponding node

l_d	τ_d	Γ	Original Data Packet

Fig. 3. Packet structure

node that has written in the packet. To prove that ELIP is more efficient than LE, we have to show that the overhead introduced by Γ is compensated by a smaller overhead generated during the lookup phase (cf. Section 4).

3.2.1 Packet Creation and Forwarding

We assume in a first time that source s has an estimation of d's position (d is the destination). After s creates the original data packet, it fills the l_d field with $\hat{l}_{s,d}$ and the field τ_d with $\tau_{s,d}$. The Γ field is filled or not depending on the criteria presented in the next paragraph. The packet is then ready to be sent using a geographic forwarding protocol. In the case where s has no location estimation of d, we have to implement a lookup algorithm. In this paper, we simply use the EASE algorithm as the lookup mechanism [8].

3.2.2 Writing Coordinates in a Packet

Let $P_{s,d} = \{s, p^1_{s,d}, p^2_{s,d}, \ldots, p^k_{s,d}, d\}$ be the path between s and d, where k is the number of nodes in the path (excepting s and d), $s = p^0_{s,d}$, and $d = p^{k+1}_{s,d}$. Node $p^i_{s,d}$ has the possibility to disseminate its location information, $\Gamma_{p^i_{s,d}}$, to every node $p^j_{s,d}$, $i < j \leq k+1$, by writing it in the Γ field of the data packets. In this paper, we assume that Γ can contain coordinates of only one node and that it is read-only, *i.e.* once a node has written its coordinates, no other node in the path can change this information. ELIP is also designed such that only a small fraction of data packets are filled with Γ, in order to keep the dissemination overhead small. The probability for a node of writing in a packet will be described in Section 3.2.4.

3.2.3 Updating Position Tables

In ELIP, a node may update its position table in three different cases. First, it uses simple encounters as in the LE scheme. Each node stores the position of the last encounter with any other nodes. Second, nodes update their position tables using information carried in the $\mathit{\Gamma}$ field of transit packets. Destination d as well as all nodes $p^j_{s,d}$, $i < j \leq k$, update the $\{\mathtt{ID} = p^i_{s,d}\}$ entry in their position table by reading the information written by $p^i_{s,d}$ in $\mathit{\Gamma}$. Third, any node in the path $P_{s,d}$ can update location information about the destination if the header of a data packet in transit contains fresher information than the one in its local position table.

3.2.4 Insertion Probability

Let π be the probability that a node write in a packet. In this paper, we consider that π is fixed and is the same for all the nodes. Let us consider a packet to be routed from source node s to destination node d through the path $P_{s,d} = \{s$, $p^1_{s,d}$, $p^2_{s,d}$, $\ldots$, $p^k_{s,d}$, $d\}$. Since we apply a read-only policy, the probability for a node p^i to write its coordinates in the $\mathit{\Upsilon}$ field is then $\pi(1-\pi)^i$.

4 Analysis

We present in this section some simulation analysis of the proposed system.

4.1 Simulator Model

We have conceived a network simulator to evaluate the efficiency of ELIP and LE in disseminating nodes' coordinates. We describe in the following the network model used in our simulations.

- **Topology.** The emulated network environment is a square universe of 1000 meters on a side, partitioned into a grid with squares of one square meter. Vertices of the grid are the positions where nodes can be placed. Nodes' initial positions are randomly chosen.
- **Neighborhood.** Each node in the network has as immediate neighbors all nodes in a range of r meters. We suppose links to be symmetric. An important parameter in our simulations is the relative density, given by the number of neighbors per coverage zone.
- **Mobility.** We use in our simulations the popular and commonly used random waypoint (RWP) mobility model. The waypoints are uniformly distributed in the area.
- **Time scale.** We assume that packets travel much faster than nodes. The topology is supposed then to be frozen during packet transfer.

- **Traffic quantity.** We have simulated both LE and ELIP for different values of traffic patterns, which is represented by the maximum number of pairs (source, destination) that communicate in parallel. This value is set as a percentage of the total number of nodes in the network. For instance, for 1000-node topology and a 10% traffic probability, the maximum number of simultaneous pairs (source, destination) is 100.
- **Forwarding algorithm.** We assume that nodes know their current coordinates and the ones of their direct neighbors. For forwarding, we use a classical geographic method. Suppose that node i, located in l_i, receives a packet to be forwarded to d, with l_d. Let $N_i = \{n_1, n_2, ..., n_l\}$ be the set of i's neighbors and $\delta(i, j)$ be the Euclidean distance between i and j. Node i forwards the packet to neighbor n_x located in l_{n_x} if $\delta(n_x, d) < \delta(i, d)$ and $\delta(n_x, d) \leq \delta(n_y, d)$, $\forall n_y \in N_i$.

4.2 Measurements

The simulator is written in ReactiveML [14], a language dedicated to the simulation of complex dynamic systems.[3] The advantages of using ReactiveML are twofold: efficient runtime and compact code. In the simulator, each node is a process that moves, discovers the neighborhood, and routes packets.

We have conducted a set of simulations in order to evaluate the efficiency of disseminating node locations with ELIP. Recall that we do not focus here on the performance of APB routing protocols, but on the advantages of using ELIP over the Last Encounter algorithm. In our simulator, each node has two position tables, one for LE and another for ELIP, and they use EASE as the lookup mechanism. This guarantees that both ELIP and LE are evaluated under identical conditions.

For each route, we measure: average age of location information, search depth, cumulative overhead, and path length.

4.2.1 Average Age of Location Information

We consider in the beginning of our simulations that each node has information only about its direct neighbors. We randomly pick up a node n and at each iteration of the algorithm, we measure the dissemination degree of its location information. The dissemination degree is estimated as the age of n's location information averaged over all nodes in the network.

We can see in Fig. 4 that, whatever the distance, the average age given by ELIP is upper bounded (with a factor of two) by the result of LE. This figure also shows that, as expected, using ELIP instead of LE allows distant nodes to have fresher information about n's position.

4.2.2 Search Depth

As described in Section 2.2, when anchor node a_i receives a message, it searches around its own position a node having fresher information about the destination

[3] The simulator is available at `http://www-spi.lip6.fr/~mandel/rml/simulator`

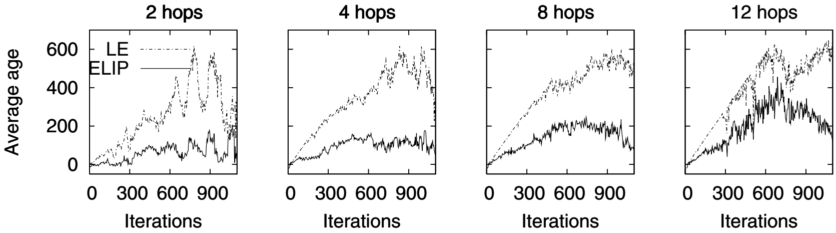


Fig. 4. Average age of n's coordinates depending on distance between n and the other nodes. The network density for this simulation is fixed to 8 nodes per coverage zone

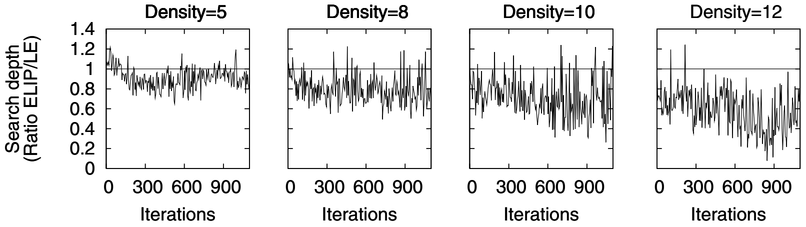


Fig. 5. Average search depth, which represents the distance in number of hops between the anchor and the node which responds with a better estimation of destination's location

than the one carried by the packet. The search depth is the distance, in number of hops, separating the anchor node and the node that responds to the local flood. In our simulations, we use EASE as the lookup algorithm for both LE and ELIP with different topology densities. Fig. 5 shows the ratio between current overheads generated by ELIP and LE. Observe that the ratio is almost all the time inferior to 1. This ratio varies between 0.5, when density is 12, and 0.9 for density 5.

4.2.3 Cumulative Overhead

For LE, traffic overhead is generated only when anchor nodes search for a better estimation of the destination's position. In ELIP, we also generate traffic overhead in data packets. It is then important to prove that, although ELIP incurs some overhead during the dissemination, the overall overhead, *i.e.* dissemination + lookup, is reduced.

Fig. 6 shows the cumulative overhead generated during the simulation. In the beginning, when the topology density is low, ELIP generates more overhead. The reason for this is that the position tables are about the same in both ELIP and LE cases. Thus, lookups lead to similar overheads. However, the overall overhead generated by ELIP is higher because of the extra overhead generated during the dissemination phase. But this overhead is rapidly compensated as data packets travel around.

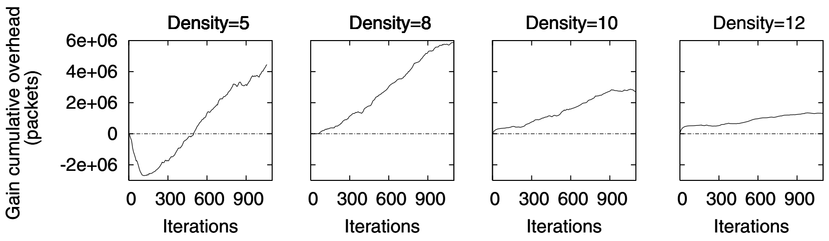


Fig. 6. Total overhead as a function of time

4.2.4 Path Length

We also performed a set of simulations in order to evaluate the routes computed when using both LE and ELIP. We can see in Fig. 7 that routes computed when using ELIP are shorter. The difference between route lengths when using ELIP and LE grows with increasing densities. We see in the same figure that, when density is 5, ELIP is closer to LE. The routes obtained by ELIP are 50% shorter when density is 12 nodes per coverage region.

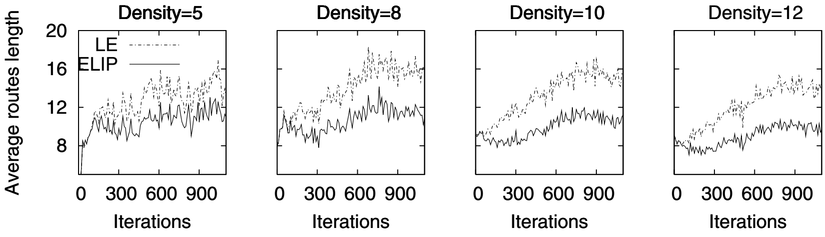


Fig. 7. Route length as a function of density

5 Conclusion

This paper proposes *Embedded Location Information Protocol*, a dissemination method for age and position based routing algorithms. Existing approaches to locate nodes in ad hoc networks justify our choice for a database-free method. In this context, ELIP is an algorithm that uses existing data packets to perform dissemination in a very efficient fashion. Through a number of simulations, we could determine the effectiveness of our approach. The results show that the performance of ELIP is always better than the simple Last Encounter method. The reason for this is that ELIP, although increasing the overhead during the dissemination phase, drastically reduces the global overhead by allowing nodes to perform lookups in smaller scopes. Future improvements of ELIP include an overwriting model for the dissemination of a larger number of location information per Γ field and an insertion probability which depends on nodes speed and mobility model.

References

[1] Capkun, S., Hamdi, M., Hubaux, J.P.: GPS-free positioning in mobile ad hoc networks. In: Proceedings of the 34th HICSS. (2001)

[2] Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low cost outdoor localization for very small devices. IEEE Personal Communications **7** (2000) 28–34

[3] Hightower, J., Borriella, G.: Location systems for ubiquitous computing. IEEE Computer Magazine **34** (2001) 57–66

[4] Kranakis, E., Singh, H., J.: Compass routing on geometric networks. In: Proceedings of 11 th Canadian Conference on Computational Geometry. (1999) 51–54

[5] Basagni, S., Chlamtac, I., Syrotiuk, V.R., Woodward, B.A.: A distance routing effect algorithm for mobility DREAM). In: Proceedings of ACM MOBICOM'98, Dallas, TX (1998) 76–84

[6] Karp, B., Kung, H.T.: GPSR: Greedy perimeter stateless routing for wireless networks. In: Proceedings of ACM MOBICOM'00. (2000)

[7] Mauve, M., Widmer, J., Hartenstein, H.: A survey on position-based routing in mobile ad hoc networks. IEEE Network Magazine **15** (2001) 30–39

[8] Grossglauser, M., Vetterli, M.: Locating nodes with EASE: Last encounter routing in ad hoc networks through mobility diffusion. In: Proceedings of IEEE Infocom. (2003)

[9] Johnson, D.B., Maltz, D.A.: DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In: Ad Hoc Networking. Addison-Wesley. (2001) 139–172

[10] Perkins, C.E., Royer, E.M.: Ad hoc on demand distance vector (AODV) routing. In: Proceedings of IEEE Wrokshop Mobile Computing systems and Applications (WMCSA'99). (1999)

[11] Viana, A.C., Amorim, M.D., Fdida, S., Rezende, J.F.: Indirect routing using distributed location information. to appear in ACM Wireless Networks (2004)

[12] Eriksson, J., Faloutsos, M., Krishnamurthy, S.: Scalable ad hoc routing: The case for dynamic addressing. In: Proceedings of IEEE Infocom. (2004)

[13] Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). Internet RFC 3626 (2003)

[14] Mandel, L., Pouzet, M.: (ReactiveML) `http://www-spi.lip6.fr/∼mandel/rml`.

Coordinated Interaction Using Reliable Broadcast in Mobile Wireless Networks

Theodore L. Willke[1,2] and Nicholas F. Maxemchuk[1]

[1] Columbia University, Dept. of Electrical Engineering,
1312 S.W. Mudd, 500 West 120th Street, New York, NY 10027
tlw24@columbia.edu, nick@ee.columbia.edu
[2] Intel Corporation, Enterprise Platforms Group, 2800 Center Drive,
M/S DP3-307,DuPont, WA 98327
theodore.l.willke@intel.com

Abstract. We consider the challenges of supporting collaborative decision making and applications requiring coordinated interaction in mobile ad hoc networks. This environment makes group coordination difficult due to relatively high packet loss and the presence of a continuously evolving network topology that leads to a changing set of participants. We describe and evaluate an application-level mobile reliable broadcast protocol (M-RBP) that provides applications with network characteristics well-suited to supporting coordinated interaction, such as reliable broadcast with data consistency and global ordering. The protocol is time-, rather than event-, driven, providing it with unique, deterministic operational characteristics that can be verified in a relatively straightforward manner.

1 Introduction

Wireless mobile computing and sensor networks may consist of nodes that desire to collaborate to find an interactive solution to a problem. These nodes can combine information from local sources, such as node location and other measurements, with critical information shared by their peers. By contributing this information to a global view of the problem space, the nodes are enabled to act coherently and achieve common goals.

One example of a coordinated interaction problem involves a lane of vehicles traveling in a group on a highway. The leading automobile may be faced with a decision to either brake aggressively or veer to avoid a serious accident. Its on-board computer may determine that hard braking is the lowest risk solution, as long as the vehicles close behind brake as well. If the on-board computer can broadcast this intention to the other vehicles and confirm their reception and compliance with the message by the critical deadline, the automobiles can rapidly decelerate and avoid an accident. To do this requires a communication protocol that supports reliable broadcast with consistent message delivery and confirmation.

Reliable broadcast and multicast for mobile ad hoc networks (MANETs) has been studied intensively in recent years [1], [2], [3], [4], [5], [6]. Many of the protocols developed drastically improve the reliability of packet delivery, but do not provide a

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1168 – 1179, 2005.

framework for data consistency (commitment and ordering). Token ring-based protocols pursued by other researchers, such as the work described in [7], [8], may ultimately support this framework; however they currently lack reliable message delivery mechanisms.

The Mobile Reliable Broadcast Protocol (M-RBP) was developed to support coordinated interaction and collaborative decision making in mobile ad hoc networks. It was initially described in [9]. The protocol guarantees that nodes able to make forward commitment progress will:

- Put messages in the same message order.
- Commit the same set of messages. With this and the previous attribute, each copy of the distributed program can reach the same conclusion.
- Be able to verify that a specific set of collaborators have received a message. By guaranteeing this, the protocol can enable coordinated operations.

In this paper, we provide precise operational guarantees and initial simulation results for M-RBP. In particular, we show that the protocol operates efficiently in the presence of arbitrary losses for a reasonably-sized group of collaborators located in a local geographical region. We also determine the network topologies for which the addition of an underlying routing or flooding protocol would be most beneficial.

The remainder of this paper is organized as follows. The problem of coordinated interaction is described in Section 2. Section 3 reviews the operational assumptions. Section 4 describes the protocol's operation. Section 5 describes the simulation methodology, and Section 6 presents the results with analysis. Section 7 concludes the paper.

2 Coordinated Interaction

In a replicated database, identical copies of a transaction commitment record are kept at more than one location, and more than one entity may submit and coordinate transactions [10], [11], [12]. We can draw an analogy between distributed database replication and coordinated interaction by noting that the requirement for maintaining a replicated, in order, transaction history is similar to the requirement of sharing a global view of the problem space in interaction problems. We desire to transform this sort of algorithm into one that presumes nothing about the stability of network resources or connections, and that is useful for general coordinated interaction applications. This distributed algorithm may carry out globally-optimized interactions by enabling entities to reliably distribute local information (e.g., sensor inputs or GPS location) to subsets of peers and gather pertinent feedback.

For nodes in a group to form a cohesive view of the problem space and coordinate their actions, they must be able to confirm that specific messages were received by specific peers, and that these peers are capable and willing to act on the supplied message content. In some cases the requester may not want an action to be carried out unless a specific set of nodes receive the message and respond appropriately. M-RBP provides this capability by using a process similar to a two-phase database commitment process.

In the simplest two-phase commitment process, shown in Figure 1a, a transaction is submitted to subordinates (each in initial state *q*) by a coordinating site, as described in [10]. To complete the first phase, the subordinates receiving, and able to commit, the transaction respond with a "*YES*" vote, move to wait state *w*, and the coordinator collects these votes. If one or more nodes respond "*NO*", or fail to respond the transaction will be aborted (state *a*). In phase two, the coordinator tabulates the votes, and, based on the quorum rule selected, will either issue a command to commit (state *c*) the message or abort it.

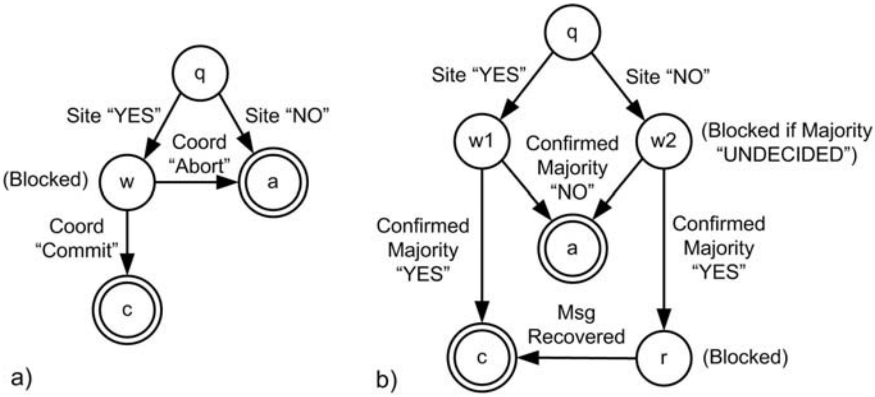


Fig. 1. a) Basic two-phase commit process (blocking) used in distributed databases, and b) M-RBP's two-phase commit process

The voting process accomplishes two things: 1) It ensures that the coordinator is made aware of potential divergences in database replications, and 2) in partitionable networks, it ensures that only one partition is committing messages at any given instant so that only one unique transaction history exists.

M-RBP takes advantage of the voting process in a modified form of the two-phase commit process that is shown in Figure 1b. In the first phase (state *q*), an elected node acknowledges the source message. Then, after a recovery period, each node individually votes "*YES*" or "*NO*" to commit the message. Using peer-to-peer communication, the nodes attempt to recover as many peer votes as possible to confirm a majority "*YES*" or "*NO*" vote. A node cannot leave the wait states, *w*1 and *w*2, until it determines a majority. Usually the decision leads to commit state, *c*, or abort state, *a*. In rare cases, the majority may vote to commit a message, whereas the individual failed to recover the message by a deadline. In this case, the node remains blocked in state *r* until it recovers the message from a peer.

The M-RBP approach to the two-phase commit process permits it to proceed in a time-driven manner, with no coordinator-slave interaction. Also, the commit quorum is a simple majority and the majority is determined by each node in a peer-to-peer manner. Nodes that block are responsible for unblocking themselves, and mechanisms are built in to the protocol to permit the group to know when a peer is blocked.

3 Operational Assumptions

Coordinated interaction is difficult in infrastructure-based IP networks, and even harder in MANETs because: 1) the network topology may continuously evolve leading, in turn, to a changing group of collaborators and 2) the wireless broadcast medium is relatively unreliable due to transmission limitations, noise, and contention. We focus in this paper on applications that involve a set of mobile collaborators residing in a geographically-localized MANET, limiting the size of the region to several transmission distances, as shown in Figure 2.

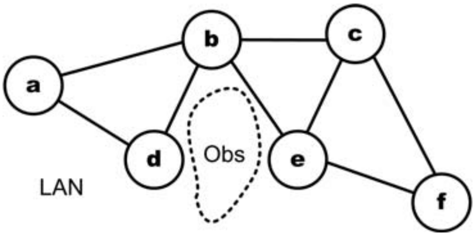


Fig. 2. A set of mobile collaborators in a small, localized, multi-hop network

M-RBP operates with the following assumptions about the nodes and the network:

- Communication links may fail or recover at any time. M-RBP was designed for the application layer and makes no assumption about the presence, or lack of, an underlying unreliable routing or flooding protocol. In its current form, M-RBP is designed to perform adequately if all control and data messages are simply broadcast to one-hop neighbors.
- The network is partitionable. One or more partitions may form temporarily or permanently. A primary partition is one in which a majority of group members reside.
- Nodes may fail and, when they do, they stop transmitting altogether. That is, Byzantine failures are not permitted.
- Messages may be received, not received, or received with CRC errors. If received with errors, the packets are dropped without notification to M-RBP.

4 Protocol Operation

An early description of M-RBP can be found in [9]. The abbreviated description of M-RBP that follows emphasizes the means by which it ensures global message ordering and consistent message commitment (or abort), as well as group membership services for the participating nodes.

The basic framework of M-RBP consists of a token ring of receivers, some of which may also be message sources. Figure 3 is a diagram of sources and receivers in the token ring, adapted from [13]. Transmitted source messages are identified by source number, s, and source sequence number, M_s. The n receivers take turns as the token site, passing an implicit, time-based token every Δ_T seconds in an order defined by a shared data structure called the *Token Passing List*. As each token site relinquishes the token, it transmits an *ACK* with a globally-unique acknowledgement sequence number. The *ACK* references the messages received during the token interval and assigns them a relative order.

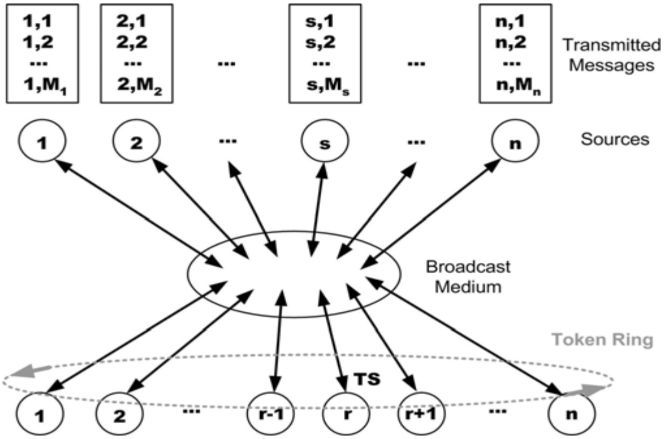


Fig. 3. Message sources transmit messages into the unreliable broadcast medium and receivers in a time-driven token ring take turns acknowledging and sequencing these messages

Beginning a short time, Δ_R, after the scheduled *ACK* transmission at $m{\cdot}\Delta_T$, where m is a positive integer, all receivers that did not receive the initial broadcast of the *ACK* request its retransmission using an *ACK Retry*. Once the *ACK* is recovered, any missing source messages are requested using a *NACK*. Retry and retransmission implosion is suppressed using mechanisms described in [9].

Starting at time $m{\cdot}\Delta_T + k{\cdot}\Delta_R$, after k potential retry rounds, scheduled *ACK*s contain information on earlier *ACK*s and source messages that could not be recovered. This information on missing *ACK*s and source messages is used as a vote to determine what *ACK*s to use for sequencing and what messages to commit.

4.1 Group Membership Service

Any node that wants to join the group requests timing and state information from a current group member and then transmits a source message including a join request. When, and if, the source message is committed, the new member is added to each node's copy of the *Token Passing List*.

To leave the group, a node may transmit a source message including a drop request. When, and if, the source message is committed, the member is removed from the *Token Passing List* and must cease transmitting M-RBP messages.

A more difficult problem is coping with receivers that fail, or are isolated, unexpectedly. To keep the timed token ring operating, the protocol requires a scheme to detect when this happens and reassign the token slots. The method chosen is to use unrecoverable *ACK*s as an indication of node failure. At a deadline peers vote, via a field in their own scheduled *ACK*s, on which *ACK*s are missing. Each member of the group uses the votes that it recovered and the following agreement function, *F*, to decide whether to remove the node in question from its copy of the *Token Passing List*:

$$F = \begin{cases} \text{Y} & \text{if } \sum \text{"Yes"} \geq \text{C(t)} \\ \text{N} & \text{if } \sum \text{"No"} > \text{C(t)} \\ \text{U} & o.w. \end{cases}, \tag{1}$$

where *C*(*t*) is one-half the number of entries on the *Token Passing List* (rounded up) at the time, *t*, that the vote ends. This agreement function returns a *Yes* (*Y*) or *No* (*N*) drop decision if a majority of the nodes voting return "*Yes*" or "*No*" votes. If a majority is not recovered, the vote remains *Undecided* (*U*). A node with an undecided vote remains blocked until it recovers the vote outcome from a peer. The node must then re-join the token ring.

The voting and agreement timeline is summarized in Figure 4.

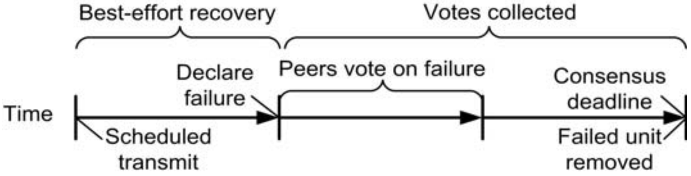


Fig. 4. Timeline for best-effort *ACK* recovery, peer voting, and peer consensus to remove a node that failed to transmit its scheduled *ACK*

4.2 Global Ordering and Consistent Commitment

The process used to build a consensus on nodes that have unexpectedly failed is also used to ensure global message ordering and consistent message commitment by each member of the group. A consensus process is required because the unreliable wireless network and best-effort packet recovery processes can result in each node recovering a different subset of the available source message and acknowledgement information.

If nodes vote and reach agreement on both the *ACK*s to use for message ordering and the source messages to commit, all non-blocked nodes can achieve a consistent global view of the message history. Furthermore, since only non-blocked nodes continue to participate in the token ring and transmit *ACKs* as the token site in the token round following a message commitment, all peers, including the message source, can verify the set of non-blocked nodes that received a message. The process timeline is summarized in Figure 5.

The first phase in the process is *L1* commit, or message injection. Sources retransmit messages until they receive an *ACK*. At time *t*, the message source, as well as other nodes in broadcast range, receives the *ACK* that references the message. Subsequently, the entire group attempts to recover and reach majority agreement on whether or not to use the *ACK* for message sequencing

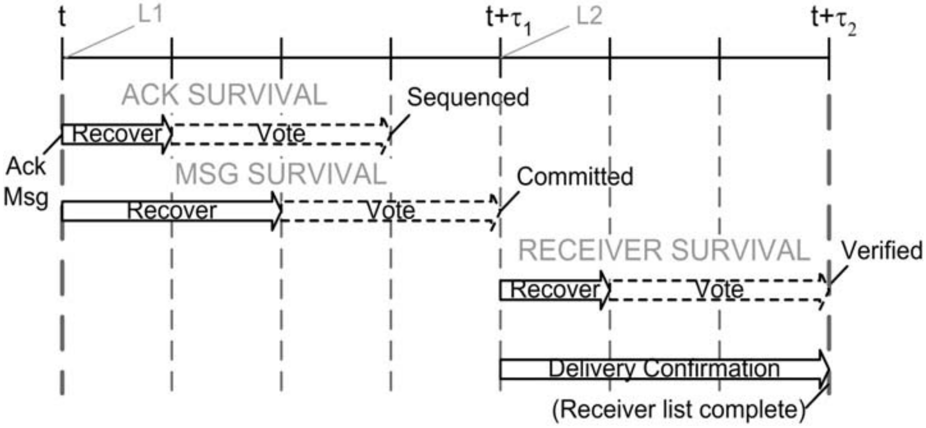


Fig. 5. *ACK* and source message commitment and receiver verification timeline

The second phase of message commitment begins with active message recovery, which starts when *ACK* recovery ends, and ends with *L2* commit, the point at which the message may be sent to the application. As with *ACK*s, a majority must recover the message and vote "*YES*" in order to use it. If this happens, the message is committed. This policy ensures that: 1) decisions are made only by a primary partition, and 2) there is a high probability that all peers in the primary partition will successfully commit the message. To maintain consistency, nodes cannot reverse a decision to commit or abort a message after reaching one.

Because the protocol is time-driven, all nodes that can reach a decision to commit will do so at a deterministic time, $t + \tau_1$, following initial message injection into the network. Nodes that are not in the primary partition, or that experience transient communication failures for an amount of time, are blocked from committing or aborting messages, and these messages remain undecided. Nodes that re-join the primary partition and retrieve vote outcome information may then disposition undecided messages.

By time $t + \tau_2$, the non-blocked members of the primary partition are able to verify the set of peers that committed the message. Sources may retransmit messages that are aborted at *L2*, or that do not reach the intended receivers.

Since a lossy broadcast network provides no FIFO guarantee for message delivery order, all nodes use the ordered set of acknowledgements to globally sequence messages. Global message ordering requires two steps. First, relative message order is assigned in each (bulk) *ACK*. Second, the *ACK*s, and their referenced messages, are ordered by increasing *ACK* sequence number, with the dropped *ACK*s and messages removed from sequencing. The earliest instance of a message reference is used; any duplicate references are ignored.

5 Simulation Methodology

Since M-RBP is the first reliable broadcast protocol for mobile wireless networks to provide global data ordering and consistency guarantees, it is impossible to quantitatively compare its performance with existing methods. However, because it is a time-, rather than event-, driven protocol, many of the simulation results are easily analyzed.

The performance characteristics of M-RBP in a MANET were evaluated using the QualNet simulator [14]. M-RBP was implemented as an application that receives data from a traffic generator application and passes control or data messages to/from other network stack layers. The supplied library models for UDP, IPv4 and the 802.11 DCF MAC [15] were also used.

The constant bit rate traffic consisted of source messages with 512 byte data payloads. The message interdeparture interval varied by experiment. The channel capacity was fixed at 2 Mb/s. The propagation range was 375 meters and a two-ray propagation model was used. The channel characteristics match those used extensively in the literature on mobile ad hoc networking.

22 nodes were randomly placed in a field size that varied by experiment. All nodes were instantiated with a copy of M-RBP and the network stack. A subset of the nodes, determined in each experiment, were chosen to be message sources and implemented copies of the traffic generator. A random waypoint mobility model with zero pause time was used and the mobility speed was varied by experiment.

6 Results and Analysis

The performance metrics we used are packet delivery ratio, additional messages per source message committed, total number of packets transmitted during the simulation, and commitment delay. Packet delivery ratio is the portion of *L1* committed messages that achieve *L2* commitment. The additional messages per source message committed is the average number of data and control messages (i.e., source message retransmissions, *NACK*s, scheduled *ACK*s, *ACK* retransmissions, and *ACK* Retries) transmitted per *L2* commit by the group. The total number of packets transmitted during the simulation is a measure of total network load. The commitment delay is the time for a *L1* committed message to reach *L2* commit, where the information can be passed to the application.

In all scenarios tested, the packet delivery ratio was 100%. Additionally, data consistency and ordering was checked across the set of receivers in the group and found to be correct in all scenarios tested.

Traffic Rate - Since M-RBP uses bulk acknowledgements and a fixed acknowledgement rate, we expect the protocol to become more efficient as traffic rate increases. This is illustrated in Figure 6, where the traffic rate is expressed as the average number of source messages transmitted per token passing interval (4 sources). High and low node density cases were simulated. In the high density case, all nodes in the group are within one hop of each other, whereas, in the low density case, nodes can reach ~20% of the remainder of the group in one hop. The high density results are very close to the lower theoretical bound, which is comprised of the average number of scheduled ACK transmissions per source message commitment.

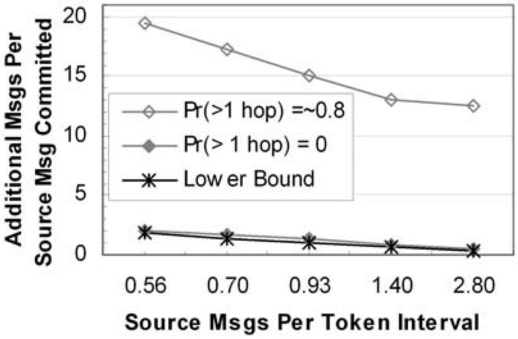


Fig. 6. Protocol efficiency as a function of traffic rate

Commitment Delay – Applications desire a low commitment delay to enable fast program reaction time. M-RBP's commitment delay is a deterministic function of the token passing rate, group size, and best-effort recovery policies. The protocol packet overhead is also a function of these parameters and the offered source message load.

To determine the relationship between protocol overhead and commitment delay, we ran a series of simulations in which the source message load was kept constant while the commit delay was varied. A relative commitment delay of 1.0 is defined as the delay that results for a group of 22 nodes, a 30 ms token passing interval, a maximum retry count of 15, and a retry period of 24 ms. Four CBR sources injected packets every 500 ms into a network of nodes moving at 30 m/s in a 750 m x 750 m field.

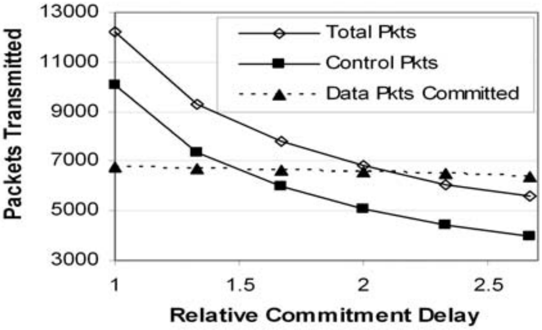


Fig. 7. Total number of data and control packets transmitted during a simulation run, as a function of the relative commitment delay

The results are shown in Figure 7 for a 40 second simulation. Note that the number of packets committed in each run is approximately the same. The number of total packets transmitted reduces exponentially as the commitment delay is relaxed, and is strongly dependent on control packet load (i.e., *NACK*s, *ACK*s, and *ACK Retries*).

Field Size – To study the effect of hidden nodes, or multi-hop networking, on protocol overhead, the field size was varied for a group of constant size. Changing the field size effectively alters the probability, Pr(> 1 hop), that any given node is greater than one hop from the original message or ACK source. For Pr(> 1 hop) = 0, the entire group is a clique. Traffic load matched that used in the commit delay experiment.

The results are shown in Figure 8. The protocol is reasonably efficient as long as at least ~40% of the nodes are within one hop of each other. This suggests that the protocol may benefit from an underlying routing or flooding algorithm in networks with larger hop radii, but that the protocol operates efficiently in small networks.

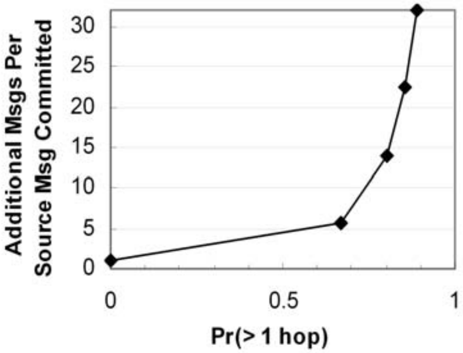


Fig. 8. The additional packet overhead required, as a function of the probability that any given group member is greater than one hop from the original message or *ACK* source

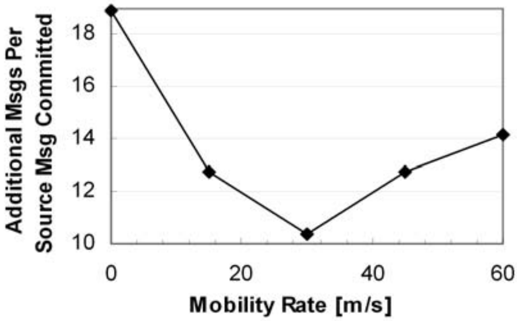


Fig. 9. Protocol packet overhead as a function of node mobility rate using a random waypoint model and zero pause time

Mobility Rate – To study the effects of node mobility rate on protocol efficiency, simulations were run at a field size of 750 m × 750 m, with the same traffic load as the commit delay experiment. Node speed was varied from 0 m/s to 60 m/s, in 15 m/s steps. Because M-RBP does not rely on topology-related state information, its performance does not degrade as mobility rate is increased. In fact, as shown in Figure 9, the protocol performs better over a wide range of terrestrial speeds when compared to the static scenario. The increased efficiency is likely due to opportunistic information spreading and increased randomness in retry and retransmission timing.

7 Concluding Remarks

We have presented the problem of coordinated interaction amongst mobile nodes in a wireless ad hoc network and described a mobile reliable broadcast protocol, M-RBP, that can provide deterministic reliability guarantees, data consistency, and ordering, to support this challenging class of application. The functionality provided by M-RBP resembles that provided by a distributed database with data replication, and is what sets it apart from other highly-reliable broadcast and multicast protocols developed for MANETs.

We have provided a detailed overview of M-RBP's operation and have investigated its characteristics using the QualNet simulator. At moderate levels of network loading, M-RBP operates with low control overhead, due its use of bulk acknowledgements, *NACK*s, and *ACK Retries*. Its time-driven nature supports deterministic commitment delays for non-blocked nodes, but there is a price to be paid for lower commitment delays, in terms of packet overhead.

M-RBP works efficiently over wide range of mobility rates pertinent to terrestrial applications. It was shown to operate efficiently without an underlying routing or flooding protocol, even when most of the group is not in direct communication with one another. In larger multi-hop networks, M-RBP should maintain its efficiency with the addition of an unreliable, state-based message forwarding layer.

References

1. Chandra, R., Ramasubramanian, V., Birman, K., Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks, Proc. IEEE ICDCS, (2001) 275-283
2. Luo, J., Eugster, P. Th., Hubaux, J.-P. , Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks, IEEE Proc. INFOCOM, (2003) 2229-2239
3. Tang, K., Gerla, M., MAC Reliable Broadcast in Ad Hoc Networks, IEEE MILCOM, (2001) 1008-1012
4. Tang, K., Obraczka, K., Lee, S.-J., Gerla, M., Reliable Adaptive Lightweight Multicast Protocol, Proc. IEEE Intl. Conf. on Comm., vol. 2, (2003) 1054-1058
5. Gopalsamy, T., Singhal, M., Panda, D., Sadayappan, P., A Reliable Multicast Algorithm for Mobile Ad Hoc Networks, Proc. ICDCS, (2002) 563-570
6. Liao, W., Jiang, M.-Y., Family ACK Tree (FAT): Supporting Reliable Multicast in Mobile Ad Hoc Networks, IEEE Trans. Veh. Tech., vol. 52, no. 6, (2003) 1675-1685

7. Malpani, N., Chen, Y., Vaidya, N.H., Welch, J.L., Distributed Token Circulation on Mobile Ad Hoc Networks, to appear in IEEE Trans. Mobile Computing, (2004)
8. Lee, D., Attias, R., Sengupta, R., Tripakas, S., A Wireless Token Ring Protocol for Ad-Hoc Networks, Proc. IEEE Aerospace Conf., (2002)
9. Willke, T., Maxemchuk, N., Reliable Collaborative Decision Making in Mobile Ad Hoc Networks, Proc. 7th IFIP/IEEE Intl. Conf. MMNS, (2004) 88-101
10. Keidar, I., Dolev, D., Increasing the Resilience of Atomic Commit, at No Additional Cost, Proc. 14th ACM Sym. Principles Database Sys., (1995) 245-254
11. Amir, Y., Danilov, C., Miskin-Amir, M., Stanton, J., Tutu, C., On the Performance of Wide-Area Synchronous Database Replication, Technical Report CNDS-2002-4, Johns Hopkins University (2002)
12. Jajodia, S., Mutchler, D., Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database, ACM Trans. Database Sys., vol. 15, no. 2, (1990) 230-280
13. Maxemchuk, N., Reliable Multicast with Delay Guarantees, IEEE Comm. Mag., vol. 40, no. 9, (2002) 96-102
14. QualNet User’s Manual, version 3.6, Scalable Network Technologies, Inc., (2003)
15. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, P802.11, (1999)

Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks

Khaled A. Harras, Kevin C. Almeroth, and Elizabeth M. Belding-Royer

Department of Computer Science, University of California,
Santa Barbara, CA 93106-5110
{kharras, almeroth, ebelding}@cs.ucsb.edu

Abstract. The incredible growth in the capabilities and functionality of mobile devices has enabled new applications to emerge. Due to the potential for node mobility, along with significant node heterogeneity, characteristics such as very large delays, intermittent links and high link error rates pose a new set of challenges. Along with these challenges, end-to-end paths are assumed not to exist and message relay approaches are often adopted. While message flooding happens to be a simple and robust solution for such cases, its cost in terms of network resource consumption is unaffordable. In this paper, we focus on the evaluation of different controlled message flooding schemes over large-scale, sparse mobile networks. We study the effect of these schemes on message delay and network resource consumption. Our simulations show that our schemes can save substantial network resources while incurring a negligible increase in the message delivery delay.

Keywords: Mobile Networks, Delay Tolerant Networking.

1 Introduction

Today's Internet, as well as various other networks, operate on some often overlooked assumptions. These assumptions include: small end-to-end Round Trip Times (RTTs), a complete end-to-end path between sources and receivers, and end-to-end reliability. Packet switching is assumed to be the right abstraction for end-to-end communication in these networks.

Currently, however, new challenges are surfacing as different kinds of applications and networks emerge, especially with the incredible growth in mobile devices. These devices are driving users' needs and demands toward the expectation of connection availability in all places and at all times. These demands are creating some formidable challenges such as large delays, intermittent end-to-end connectivity and high link error rates. In essence, the underlying network architecture and protocols are becoming increasingly heterogeneous with wide variations in performance characteristics and capabilities. Applications that have

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1180–1192, 2005.

these challenges include satellite networks, planetary and interplanetary communication, military/tactical networks, disaster response, and various forms of large-scale or sparse ad-hoc networks.

These new applications and challenges, along with the impossibility of ubiquitous deployment of a fixed wired Internet infrastructure, have stimulated a great deal of research. Research in the areas of Mobile Ad Hoc Networks (MANETs) [1],[2],[3],[4],[5],[6], disconnected sparse networks [7],[8],[9],[10],[11],[12] and Delay Tolerant Networks (DTNs), [13],[14], has addressed some of these challenges and problems. However, despite the great contribution of these areas, many issues require further research. This can be demonstrated by the following scenario: A person is driving on a highway in southeastern California, carrying his own laptop or PDA, and needs to send an urgent email or submit a transaction. There is no nearby connectivity (in a desert for instance). The user passes other cars, buses or trains that have other people carrying similar devices. These users can serve as relays for the email or transaction and pass it on to others. Eventually, the message reaches someone with Internet connectivity and a direct path to the destination.

This scenario is an example of what we call a *large-scale sparse mobile network*. Most research fails to solve many problems posed by such scenarios. MANETs, for instance, focus more on networks where an end-to-end path is assumed to exist. On the other hand, partially-connected and sparse networking research usually assumes some sort of control over nodes in the network, a large degree of homogeneity, or some degree of knowledge that nodes must carry regarding other nodes in the network (e.g. the path or route a node will take). Clearly these assumptions do not hold in our scenario. DTN research comes closest to addressing the problems that arise from the scenario mentioned above. Our work complements that of the DTN community.

In this paper, we study the impact of controlled message flooding schemes over sparse mobile networks on message delay and network resource consumption. We examine the use of a probabilistic function for message forwarding. We then add a time-to-live (TTL) or kill time value on top of the probabilistic function. Finally, we add the idea of a Passive Cure on top of the other schemes and see what effect it has on the network. The Passive Cure is basically used to "heal" the "infected" nodes in the network.

The rest of this paper is organized as follows. Section 2 reviews areas of research related to our work. Details of our architecture, assumptions and the various controlled flooding schemes are presented in Section 3. Section 4 describes our simulation environment. Section 5 then discusses the results of our simulations. Finally, the paper is concluded in Section 6.

2 Related Work

Research in the areas of MANETs, disconnected sparse mobile networks and delay tolerant networks have tackled issues related to the new challenges and assumptions stated in Section 1. We briefly present some of the solutions that have been proposed to solve some of these challenges.

Much of the research on MANETs has focused on routing, introducing various proactive, reactive or hybrid routing protocols that try to find a path from a source to a destination [1],[2],[3],[4],[5],[6]. All of these routing algorithms, however, assume the existence of an end-to-end path. In other words, if there is no route, no communication can occur, and there is no attempt to relay the message. Due to this, these protocols are unlikely to find routes in sparsely connected networks. Our work specifically deals with cases in which this assumption is likely not to be true.

With the growing realization that there are many cases where an end-to-end connection cannot be assumed, several solutions have been proposed. All of these solutions rely on some form of store-and-forward relaying of messages [7],[8],[9],[10],[12],[11]. However, these solutions make assumptions that cannot be applied to the scenario proposed in Section 1. Some, for instance, assume control over node movement [7]. Others assume knowing the path that some nodes will take, as well as the time at which the node will take that path, as in message ferrying [12]. Finally, some consider only static nodes as in Data Mules [10]. Epidemic Routing [8] avoids such assumptions by simply flooding the network. The problem, however, is that Epidemic Routing creates a significant traffic load that consumes network resources and does not scale well. Our work adds to that of Epidemic Routing by finding ways to control this network flood and resource consumption.

Finally, within the Internet Research Task Force (IRTF), the Delay Tolerant Networking Research Group (DTNRG)[1] evolved from the Inter-Planetary Networking Research Group (IPNRG)[2]. The group focuses on the construction of a complete architecture that supports various protocols to achieve connectivity among heterogeneous networks in extreme environments [15],[13]. Routing issues in such environments have recently been investigated as well [14]. Our work complements that of the DTNRG by looking at a special case within the general architecture they present [13]. We also fill in the gap of looking at routing issues (through controlled flooding schemes) in sparse mobile networks in particular, and look at cases where no knowledge "oracles" [14] are assumed to exist.

3 System Architecture

This section describes our proposed architecture. We first introduce our assumptions, and briefly explain components of our architecture, along with the functionality and behavior of each component. Following that, we introduce the notion of *willingness*, which is a reflection of the degree at which nodes are willing to participate in relaying messages. Finally, we present the different schemes we use to control message flooding.

[1] http://www.dtnrg.org/

[2] http://www.ipnsig.org/

3.1 Basic Architectural Components and Assumptions

To satisfy the problems and challenges posed by the scenario mentioned in Section 1, we propose a transport layer overlay architecture for sparse mobile networks. Message forwarding and handling is done by this overlay layer and handles the heterogeneous protocols and node characteristics. This approach also complies with the basic DTN architecture [13].

There are two important assumptions made in our system. The first is **Node Blindness**, where the nodes in the network do not know any information regarding the state, location or mobility patterns of other nodes. The second is **Node Autonomy**, where each node has independent control over itself and its movement. The reason behind these assumptions is to closely model the real world scenarios described earlier. When driving through a sparse environment, any given node has no knowledge of other nodes that come within its range (Blindness), and it is autonomous in its movement (Autonomy).

In Figure 1 we show that the nodes in the network are divided into three types. A **Sender Node "S"** is the node that initiates the transmission of a message to a destination in the network. A **Forwarder Node "F"** is any node that carries the message from the sender, or another forwarder, with the aim of relaying it to the ultimate node. Finally, the **Ultimate Node "U"** is basically the final destination.

The basic mechanism of node interaction is shown in Figure 1. The interaction of nodes is similar to that in Epidemic Routing [8], where each node continuously tries to relay the message to other nodes within range that do not already have the message. We look at an example where a sender node, *S*, needs to send a message.

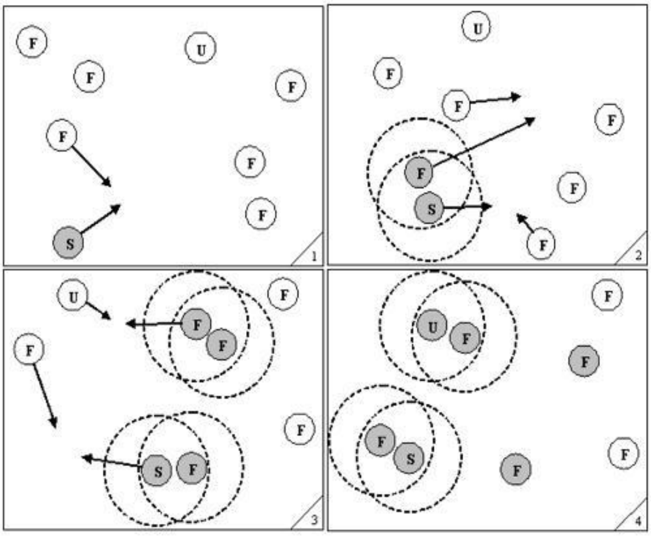


Fig. 1. Message propagation over a sparse mobile network. Shaded nodes are those carrying a copy of the message. Arrows indicate the direction of movement

At this point, node S initiates a periodic beacon for neighbor discovery purposes, where it announces that it has a message that needs to be forwarded to a certain destination. When S comes within range of one or more forwarder nodes F (or even the ultimate node U), the beacon is received and an ack is sent to S from each node that received the beacon and does not have a copy of the message. When S receives an ack for its beacons, it simply broadcasts the message to its neighbors. Once the message is received, the forwarder node starts to act as a sender node. It sends its own beacons, and both nodes travel through the network looking for either another forwarder to pass the message on, or for the ultimate node. The message gradually propagates through the network until it eventually reaches the ultimate node. This process results in the overuse of network resources through continuous and repetitive flooding of messages. On the other hand, the advantage of this approach is the high delivery rate and small delay.

3.2 Modelling Node Willingness

Generally speaking, the frequency at which a sender or forwarder node tries to forward a message depends on many factors. Some of these factors include node state (power or buffer space, for instance) or message state (size or priority of message). We model this as the **Willingness** of a node. The willingness of a node is the degree at which a node actively engages in trying to re-transmit a message. Willingness can be modelled in terms of three main variables. First, the **Beacon Interval** is the amount of time a sender or forwarder node waits before sending a new beacon. Second, the **Times-to-Send** is the number of times a node successfully forwards a message to other nodes in the network before it stops forwarding the message. Third, the **Retransmission Wait Time** is the amount of time a node waits without beaconing before it tries to resend the message to other nodes in the network. The source node includes the value of these parameters as part of the message header. This way, the forwarder nodes can set their willingness levels accordingly.

To help clarify how these three variables affect the behavior of a given node in the network, we introduce the following simple example. Let us assume that the beacon interval = 1 sec, times-to-send = 2, and the retransmission wait time = 50 sec. This means that when a sender wants to send a message, it sends a beacon every second to find other nodes that would carry the message. Once the sender node finds a forwarder node (by receiving an ack for its beacon), it transmits the message and decrements the times-to-send by one. The sender then waits for 50 seconds before it resumes sending beacons every second to look for the next node to which to forward the message. This process repeats until the times-to-send reaches zero. The forwarder nodes that received the message in both cases start acting as the original sender (assuming that all nodes have the same willingness).

3.3 Specific Controlled Flooding Schemes

In this paper, we study different controlled flooding schemes. We are careful to base our schemes on simple, non-chatty and elegant algorithms. This is because in sparse and highly mobile networks, complex or chatty algorithms waste the short time nodes have when they come within range of each other. The schemes we introduce are the following:

1) Basic Probabilistic (BP): When talking about the *willingness* of a node in the previous section, we implied that forwarder nodes have the same willingness as the sender node. To more closely emulate reality, however, we choose a uniform distribution probabilistic function that determines the willingness of the nodes to transmit a given message. Based on the result of this function, a forwarder may choose not to forward the message at all, forward it at half the willingness of the sender or forward it at the same level of willingness as the sender.

2) Time-to-Live (TTL): In this scheme, we add a time-to-live value. The TTL here determines how many times the message is forwarded before it is discarded. We add the TTL on top of the BP scheme since the BP scheme is a more realistic representation of how nodes act regarding the choice of forwarding messages.

3) Kill Time: Here, we add a time stamp to the message on top of the BP scheme. The time stamp is the time interval after which the message should no longer be forwarded. This is an absolute universal life-time for the message. This could be very useful if the sender node knows how long it will be disconnected. This is also a good way to set the maximum time a node should keep a message in its buffer if the times-to-send (TTS) variable of that message does not reach zero.

4) Passive Cure: The final scheme or optimization we introduce is a Passive Cure. The idea is that once the destination (Ultimate node) receives the message, it generates a Passive Cure to “heal” the nodes in the network after they have been “infected” by the message. The ultimate node “cures” the forwarder that passed the message to it by sending a cure-ack instead of a normal ack that is sent when a beacon is received. Now whenever that forwarder or the ultimate destination detect any other node sending that same message, they send a cure-ack to that node to prevent future retransmissions.

4 Simulation Environment

The main goal for our simulations in this paper is to compare the performance of the schemes described in Section 3 and study their impact on our metrics.

To start, we first describe our simulation environment. We conducted our simulations using the GloMoSim network simulator. Since we do not have real data describing our targeted scenarios, we use the random way-point model since, intuitively, it most closely approximates the scenarios we are concerned with. The node speed ranges between 20 to 35 meters per second (to get a range of almost 45 to 80 miles per hour); the rest period is between 0 and 10 seconds. Every point in our results is taken as an average of ten different seeds. We added to the simulator an overlay layer that handles the storing and forwarding of

Table 1. Simulation Parameters

Parameter	Value Range	Nominal Value
Number of Nodes	10 to 250	100
Terrain	$10km^2$ to $50km^2$	$10km^2$
Simulated Time	1hour to 24 hours	6 hours
Transmission Range	250m	250m

messages, as well as the enforcement of the various stopping techniques that we evaluate. The parameters used in our simulations are summarized in Table 1.

Other important parameters in our simulations are:

- **Beacon Interval:** Time period after which a beacon is sent. It ranges from 0.5 sec to 50 sec with a nominal value of 1 sec.
- **Times-to-Send (TTS):** Number of times to successfully forward a message. It ranges from 1 to 50 with a nominal value of 10.
- **Retransmission Wait Time (RWT):** Time period after which a node tries to resend a message. It ranges from 0 sec to 500 sec with a nominal value of 50 sec.
- **Time-to-Live (TTL):** The number of hops after which a message is discarded. It ranges from 2 to 10 with a nominal value of 7
- **Kill Time:** The absolute time after which the message is discarded. It ranges from 1000 sec to 21600 sec (6 hours, the nominal simulation duration time) with a nominal value of 5000 sec.

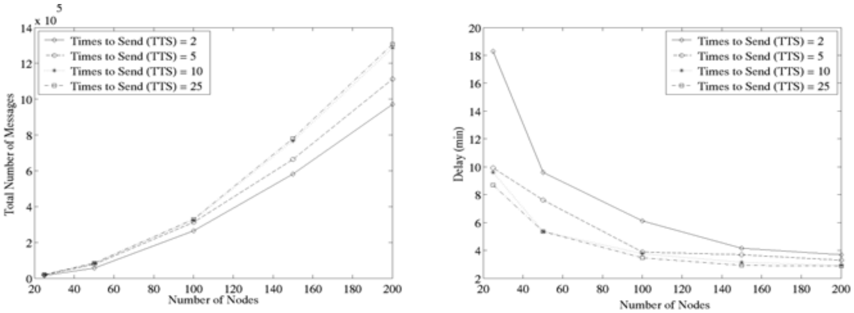
We consider two metrics in evaluating our controlled flooding schemes. First, **Network Efficiency** is represented by the total number of messages sent by the nodes in the network. Second, **Overall Delay** is the total time that elapses from when a node wants to send a message until the intended destination (ultimate node) receives that message for the first time.

5 Results

A representative set of our simulation results is presented in this section. All the results are shown for one sender node trying to send one message to a single ultimate destination. We hope to achieve two main goals with our study. First, we hope to better understand how a controlled flooding based overlay network behaves in general. Second, we want to know how the specific flooding schemes we are testing affect the network efficiency and overall delay.

5.1 Basic Behavior

Before applying our probabilistic scheme, we analyze how the network acts assuming full willingness of all the nodes in the network along with enforcing some basic level of message control. This scheme is very similar to the Epidemic



(a) (b)

Fig. 2. Impact of changing the number of nodes and times-to-send (TTS) on the total number of messages (a) and overall delay (b) in the basic scheme

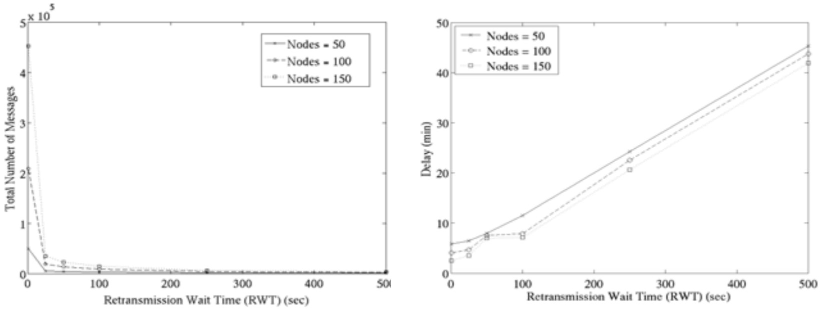
Routing scheme but with some control over message forwarding. We use the times-to-send (TTS) variable to represent node willingness. We investigate how varying the TTS and the network density affect our metrics. The retransmission wait time (RWT) in these experiments is 1 sec.

Figure 2(a) shows the total number of messages sent by all the nodes in the network and Figure 2(b) shows the delay. Overall, Figure 2 shows that increasing the density of the network results in a large increase in the total number of messages sent in the network. One interesting point is that an increase in the network density results in a significant decrease in the overall delay only up to a certain point (Number of nodes = 100), after which the decrease in delay that we achieve is overshadowed by the corresponding increase in cost. On the other hand, increasing the willingness beyond a certain point (TTS = 10) does not have a large effect on either metric. The reason for this is that the nodes in such sparse networks do not encounter each other often enough to consume the large value of the times-to-send (TTS).

5.2 Basic Probabilistic Behavior

After applying our basic probabilistic scheme, we examine how the network density, retransmission wait time (RWT) and times-to-send (TTS) impact our metrics. We assume that 25% of the nodes in the network have zero willingness, 25% have full willingness and 50% of the nodes forward the message with only half the willingness (i.e. half the TTS of the source node).

Figure 3 shows the result of varying the network density while keeping the TTS set to 10. One interesting observation is the significant drop in terms of the total number of messages when the RWT increases, with only a corresponding small increase in delay (until RWT reaches 100). With a small RWT, the message spreads rapidly through the network, and in a very short time, many forwarders are actively trying to send the message. As the RWT increases, the message does not initially reach as many forwarders. This results in a significant drop



(a) (b)

Fig. 3. Impact of changing the retransmission wait time (RWT) and number of nodes on the total number of messages (a) and overall delay (b)

in the number of messages. Also, with a large RWT, the TTS is not consumed as quickly, so a forwarder node stays in forwarder mode longer, thus rejecting receipt of the same message. With a small RWT, the TTS is quickly consumed. Because the nodes do not keep any state or cache, they are ready to receive the same message and start forwarding it again.

It is important to note here that similar patterns are generated when keeping the number of nodes constant, while varying the TTS. We have also seen similar patterns when we used only the basic controlled scheme, but with a much larger scale in terms of the total number of messages. This shows how a uniform based probabilistic scheme performs much better. The results are not shown due to space limitation.

5.3 Time-to-Live (TTL)

With the improvement in the basic probabilistic scheme, we build the other schemes on top of it to see their impact on our metrics. Figure 4 shows the impact of adding TTL only and adding TTL + Passive Cure, to the basic probabilistic (BP) scheme. We discuss the Passive Cure results later and here we focus on the TTL.

The first result we observe in Figure 4 is the large decrease in the total number of messages sent in the network (note the scale is significantly smaller than in figures 2(a) and 3(a)). This is because the TTL puts a limit on the number of times a message is forwarded. Previously, messages endlessly propagated throughout the network with no limit other than the nodes' willingness.

The impact of TTL on delay is not show here. However, we found that as the TTL increases, the overall delay decreases up to a certain point (at TTL = 7), after which it stays constant at 10 minutes. It is important to note that even though there is a probability of message loss when using low TTLs, we think it is a better choice when added to the basic probabilistic (BP) scheme. If a large

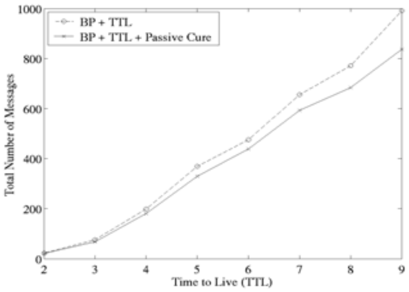


Fig. 4. Impact of adding TTL and Passive Cure to BP

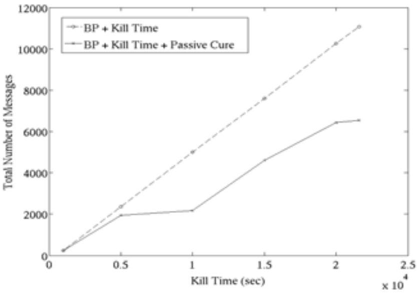


Fig. 5. Impact of adding Kill Time and Passive Cure to BP

TTL is set, our simulations show that all messages reach the destination, and the cost is much smaller than in the BP scheme. For instance, if we choose a TTL of 9, we incur a cost of 1000 messages instead of 25000 messages while keeping the overall delay is the same in both cases.

5.4 Kill Time

Figure 5 shows the impact of adding kill time only, and adding kill time + Passive Cure, to the basic probabilistic (BP) scheme. Here we focus on the kill time results, leaving the Passive Cure for later.

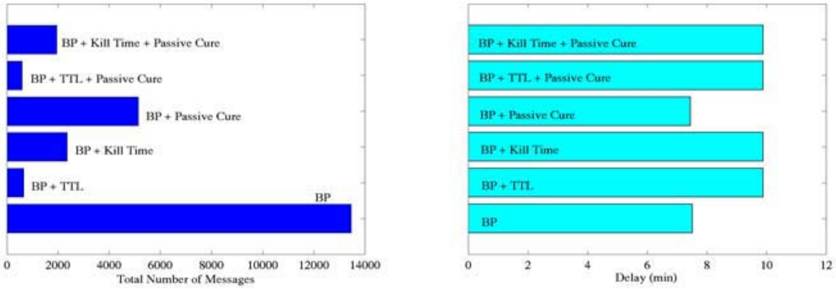
The kill time scheme introduces another improvement over the basic probabilistic (BP) scheme alone. The use of a universal time after which a message is discarded certainly stops message transmission and propagation. Figure 5 shows how the total number of messages increases as the kill time is increased. On the other hand, the overall delay (not shown due to space limitation) remains constant at 10 mins, unless the kill time is set too small that some of the messages do not make it to the destination.

5.5 Passive Cure

In Figures 4 and 5, we demonstrate the effect of adding the Passive Cure optimization to the TTL and kill time, respectively. The Passive Cure does not introduce any improvement in terms of delay because it does not help the message get to the ultimate destination any faster.

The effect of the cure is evident only in the total number of messages sent in the network. As Figures 4 and 5 show, when the Passive Cure is added, there is a drop in the total number of messages. This is due to the fact that when the cure starts to operate, the cured nodes stop trying to forward the messages even if the kill time has not been reached or if the TTL has not been consumed.

It is certainly worth mentioning that the Passive Cure optimization has several advantages over the other schemes. First, if the message reaches the ultimate node early, little network flooding may take place since the cure stops the flood early on. Second, the ultimate node receives the message only once. In all the



(a) (b)

Fig. 6. Comparing the impact of the controlled flooding schemes on the total number of messages (a) and the overall delay (b)

other schemes, the ultimate node may receive the same message multiple times. Finally, the cure may be used as a way to implement end-to-end acknowledgments if it propagates back to the sender node, since the sender would then know that its message reached the ultimate destination.

5.6 Comparing Schemes

In this section, we compare the performance of the flooding schemes. Overall, Figure 6 shows the performance of these schemes relative to each other. When looking at Figure 6(a), we observe that the Basic Probabilistic scheme by itself is the most expensive, while the basic probabilistic (BP) + TTL + Passive Cure is the least expensive in terms of the total number of messages sent in the network. Note that the basic probabilistic (BP) scheme already performs better than the basic flooding technique (which is analogous to Epidemic Routing).

Figure 6(b) shows that most of the schemes have similar overall delay. The advantage is for the basic probabilistic (BP) and basic probabilistic (BP) + Passive Cure schemes. The reason for this is that the complexity introduced by the TTL or the kill time results in an overall increase in delay. However, a two minute increase, with a corresponding large decrease in number of messages and beacons sent, is certainly acceptable.

Figure 6 summarizes the general tradeoffs between the schemes we introduce. For example, adding the Passive Cure to the basic probabilistic (BP) scheme saves about 60% of the total number of messages, with no increase in the overall delay. When adding the TTL to the basic probabilistic (BP), the total number of messages is reduced by more than 90%, while the overall delay increased by only two minutes. This increase in delay does not notably affect most of the applications we envision for this work.

6 Conclusions

In this paper we study the problem of efficient message delivery in delay tolerant sparse mobile networks. We propose several controlled flooding schemes on top of a transport layer overlay architecture. The specific schemes we examine are basic probabilistic, time-to-live, kill time and Passive Cure. We study the impact of these schemes on network efficiency and overall message delivery delay. Our simulations show that for a given sparse mobile network, the schemes significantly reduce the total number of messages and beacons sent in the network. This occurs with either no increase or only a small affordable increase in the overall message delay.

Our future work includes the transmission of a message to multiple destinations. Also, since flooding based schemes do not perform well in dense environments, we need to add measures to help the nodes modify their behavior when they enter densely populated areas.

References

1. Johnson, D., Maltz, D.: Dynamic source routing in ad hoc wireless networks. In: Mobile Computing. Volume 353. (1996)
2. Perkins, C.: Ad-hoc On-Demand Distance Vector Routing. In: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA (1999) 90–100
3. Haas, Z., Pearlman, M.: The Performance of Query Control Schemes for Zone Routing Protocol. In: ACM SIGCOMM, Vancouver, Canada (1998) 167–177
4. Royer, E., Toh, C.: A Review of Current Routing Protocols for Ad-hoc Mobile Wireless Networks. IEEE Personal Communications Magazine **6** (1999) 46–55
5. Perkins, C., Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In: ACM SIGCOMM, London, England (1994) 234–244
6. Ko, Y.B., Vaidya, N.: Location-Aided Routing (LAR) in Mobile Ad hoc Networks. In: ACM MobiCom, Dallas, TX (1998) 66–75
7. Li, Q., Rus, D.: Sending Messages to Mobile Users in Disconnected Ad-Hoc Wireless Networks. In: ACM MobiCom, Boston, MA (2000) 44–55
8. Vahdat, A., Becker, D.: Epidemic Routing for Partially Connected Ad Hoc Networks. Technical Report CS-200006, Duke University (2000)
9. Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L., Rubenstein, D.: Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences With ZebraNet. In ASPLOS (2002)
10. Shah, R., Roy, S., Jain, S., Brunette, W.: Data MULEs: Modeling a Three-Tier Architecture for Sparse Sensor Networks. In IEEE SNPA Workshop (2003)
11. Zhao, W., Ammar, M.: Proactive Routing in Highly-Partitioned Wireless Ad Hoc Networks. In: The 9^{th} IEEE International Workshop on Future Trends of Distributed Computing Systems, San Juan, Puerto Rico (2003)
12. Zhao, W., Ammar, M., Zegura, E.: A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In: MobiHoc, Tokyo, Japan (2004)

13. Fall, K.: A Delay-Tolerant Network Architecture for Challenged Internets. In: ACM SIGCOMM, Karlsruhe, Germany (2003)
14. Jain, S., Fall, K., Patra, R.: Routing in a Delay Tolerant Network. In: ACM SIGCOMM, Portland, Oregon (2004)
15. Cerf, V., Burleigh, S., Hooke, A., Togerson, L., Durst, R., Scott, K., Travis, E., Weiss, H.: Interplanetary Internet (IPN): Architectural Definition. IETF Internet Draft, draft-irtf-ipnrg-arch-00.txt (work in progress) (2001)

Application Signal Threshold Adaptation for Vertical Handoff in Heterogeneous Wireless Networks

Ben Liang[1], Ahmed H. Zahran[1], and Aladdin O.M. Saleh[2]

[1] Department of Electrical and Computer Engineering, University of Toronto
[2] Wireless Technology Planning, Bell Canada

Abstract. In heterogeneous wireless systems, the seamless and efficient handoff between different access technologies (vertical handoff) is essential and remains a challenging problem. The co-existence of access technologies with largely different characteristics results in handoff asymmetry that differs from the traditional intra-network handoff problem. In the case where one network is preferred, the vertical handoff decision should be carefully executed, based on the wireless channel state, network layer characteristics, as well as application requirements. In this paper, we present an adaptive preferred-network lifetime-based handoff strategy, and investigate the effect of an application-based signal strength threshold on the signalling load, available bandwidth, and packet delay. We propose an analytical framework to evaluate the performance of the converged system. We show how the proposed analytical model can be used to provide guidelines for the optimization of vertical handoff in the next generation integrated wireless networks.

1 Introduction

Wireless technologies are evolving toward broadband information access across multiple networking platforms, in order to provide ubiquitous availability of multimedia applications. Recent trends indicate that wide-area cellular networks based on the 3G standards and wireless Local Area Networks (WLANs) will co-exist to offer multimedia services to end users. By strategically combining these technologies, a converged system can provide both universal coverage and broadband access. Therefore, the integration of heterogeneous networks is expected to become a main focus in the development toward the next generation wireless networks [1, 2, 3].

Mobility management is a main challenge in the converged network [4, 5]. Both intra-technology handoff and inter-technology handoff take place. Intra-technology handoff is the traditional Horizontal Handoff (HHO) process in which the mobile terminal hands-off between two Access Points (AP) or Base Stations (BS) using the same access technology. On the other hand, inter-technology handoff, or Vertical Handoff (VHO) occurs when the MT roams between different access technologies. The main distinction between VHO and HHO is symmetry.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1193–1205, 2005.

While HHO is a symmetric process, VHO is an asymmetric process in which the MT moves between two different networks with different characteristics. This introduces the concept of a preferred network, which is usually the underlay WLAN that provides better throughput performance at lower cost, even if both networks are available and in good condition for the user.

There are two main scenarios in VHO: moving out of the preferred network (MO) and moving into the preferred network (MI) [6]. In either case, it is highly desirable to associate the MT with the preferred network, as long as the preferred network satisfies the user application. This can improve the resource utilization of both access networks, as well as improve the the user perceived QoS. Furthermore, handoff should be seamless with minimum user intervention, while dynamically adapting to the wireless channel state, network layer characteristics, and application requirements.

In this work, we present an adaptive VHO algorithm which takes into consideration the wireless signal strength, handoff latency, and application QoS and delay tolerance. Furthermore, it can satisfy the system handoff signalling load, as well as different application requirements by the tuning of an application specific signal strength parameter. We further propose an analytical model to evaluate the performance of adaptive VHO, which is then applied to show how the VHO decision can be optimized based on multiple conflicting criteria.

2 Related Work

The traditional HHO problem has been studied extensively in the past. Several approaches have been considered in cellular networks using the Received Signal Strength (RSS) as an indicator for service availability from a certain point of attachment, as well as for comparison between the current point of attachment and the candidate points of attachment [7]:

- RSS: handoff takes place if the RSS of the candidate point of attachment is larger than the RSS of the current point of attachment ($RSS_{new} > RSS_{current}$).
- RSS plus threshold: handoff takes place if the RSS of the candidate point of attachment is larger than the RSS of the current point of attachment and the RSS of the current point of attachment is less than a certain pre-defined threshold T ($RSS_{new} > RSS_{current}$ and $RSS_{current} < T$).
- RSS plus hysteresis: handoff takes place if the RSS of the candidate point of attachment is larger than the RSS of the current point of attachment with a pre-defined hysteresis margin H. ($RSS_{new} > RSS_{current} + H$).
- A *dwell timer* can be added to any of the above algorithms. In this case, the timer is started when one of the above conditions is satisfied, and the MT performs a handoff if the condition is satisfied for the entire dwell timer interval.

For the VHO process, the RSS's of heterogeneous networks are not comparable due to the asymmetric nature of the handoff problem. However, they can be used to determine the availability as well as the condition of different networks. On the one hand, the MI decision should be based on the availability of the WLAN, in satisfactory condition, as well as the user preferences according to predefined policies [8]. On the other hand, the MO decision should maintain efficient utilization of the wireless resources, which implies reliance on the WLAN as far as the network can provide satisfactory service to the user.

As far as we are aware, there are few existing works discussing VHO beyond direct extensions to the common techniques for HHO. Two general directions for VHO algorithms are recorded in the literature. One is based on the traditional strategies of using the RSS that may be combined with other parameters such as network loading. In [9] the authors use the dwell timer as a handoff initiation criterion for their algorithm to extend the residence time of the MT in the WLAN. They combine simulation and analysis to show that associating the MT with the WLAN for the longest possible time result in user throughput improvement even during the transition period in which the RSS oscillates around the receiver sensitivity level. However, they did not define a clear mechanism for choosing the dwell timer value. In [10], Ylianttila et al. present an algorithm that adapts the dwelling timer according to the available data rates in both networks that are defined by the standards. In [11], the same analytical framework of [9] is extended to include multiple radio network environments. Their main results show that the effect of the handoff delay seems to be dominant even with the optimal choice of the dwell timer as in [10]. In [12], Zhu and McNair present two cost-based policies for VHO, which considers the available bandwidth and RSS of the available networks. The collective handoff policy estimates one cost for all services, while the prioritized multi-network handoff policy estimates the cost for each service independently.

The second approach uses artificial intelligence techniques [13, 14, 15] such as fuzzy logic and neural networks. It is worth mentioning that some of the artificial intelligence based algorithms are complex and may not be easy to implement in practical systems. It is possible to extend our work to include improvement using similar artificial intelligence approaches. However, this is outside the scope of this paper and will be left for future work.

3 Application Life-Time Adaptation

3.1 System Model

We study the overlapping of 3G cellular and WLAN networks. The cellular network is assumed to provide universal coverage, while WLAN availability is indicated by the presence of the WLAN beacons [14] that are periodically transmitted by the WLAN AP's. Mobile-IP is assumed for mobility management.

We assume that WLAN hot spots implement loosely coupled connection [16] with the 3G network using WLAN gateways. These gateways perform several tasks including serving as Mobile-IP agents and possibly providing QoS in the form of multiple service classes defined within the WLAN. However, it is worth mentioning that end-to-end QoS support requires other mechanisms such as differentiated services to be implemented over the entire network path. The details of such implementation is unimportant to the proposed VHO algorithm and mathematical analysis. We are mainly concerned about the resultant VHO delay values.

The MT is equipped with dual interfaces that allow it to communicate with both networks. However, since Mobile-IP provides only one IP tunnel, the MT can connect to one network only at a time. In addition, multi-interface mobility client software is installed on the MT. This software performs Mobile-IP signaling with the foreign and home agents. It periodically scans the available interfaces and measures the observed RSS. Then it intelligently selects the best access network according to the predefined VHO algorithm.

Within the WLAN, a log-linear path loss channel propagation model with shadow fading is used [17]. The RSS is expressed in dBm as $RSS = P_T - L - 10n\log(d) + f(\mu, \sigma)$, where P_T is the transmitted power, L is a constant power loss, n is the path loss exponent and usually has values between 2 - 4, d represents the distance of the MT from the AP, and $f(\mu, \sigma)$ represents shadow fading which is modelled as Gaussian with mean $\mu = 0$ and standard deviation σ with values between 6-12 dB depending on the environment. We assume that when the RSS is below a certain interface sensitivity level, α, the MT is unable to communicate with the AP.

3.2 Adaptive Preferred-Network Life-Time Vertical Handoff

For the MO scenario when the MT is within a WLAN, we use the RSS to estimate the expected duration after which the MT is unable to maintain its connection with the WLAN. We take into consideration the handoff delay due to MIP tunnelling, authentication, and service initiation. We further consider an Application Signal Strength Threshold (ASST), which is the required level of RSS for the active application to perform satisfactorily.

The ASST is an application dependent parameter which represents a composite of the channel bit error rate, application error resilience, and application QoS requirements. We present here how the ASST can be incorporated into the VHO decision. We further discuss in the next section how the ASST can be adjusted to optimize the overall system performance.

In discrete time, the RSS is expressed as $RSS[k] = \mu_{RSS}[k] + N[k]$, where k is the time index, $\mu_{RSS}[k] = P_T - L - 10n\log(d[k])$, and $N[k] = f(\mu, \sigma)$. The averaged RSS, $\overline{RSS}[k]$, can be estimated using a moving average $\overline{RSS}[k] = \frac{1}{W_{average}} \sum_{i=0}^{W_{average}-1} RSS[k-i]$. The RSS rate of change, $S[k]$, can be obtained by

$$S[k] = \frac{M_1[k] - M_2[k]}{W_{slope} T_{sampling}} \ , \tag{1}$$

where $M_1[k] = \frac{2}{W_{slope}} \sum_{i=0}^{i=\frac{W_{slope}}{2}-1} \overline{RSS}[k - W_{slope} + 1 + i]$, and $M_2[k] = \frac{2}{W_{slope}} \sum_{i=\frac{W_{slope}}{2}}^{i=W_{slope}-1} \overline{RSS}[k - W_{slope} + 1 + i]$.

Then, we estimate the MT lifetime within the WLAN, $EL[k]$, as follows.

$$EL[k] = \frac{\overline{RSS}[k] - \gamma}{S[k]}, \tag{2}$$

where γ denotes the ASST. Thus, $EL[k]$ represents the application specific time period in which the WLAN is likely to remain usable to the MT.

Based on the measured and estimated parameters, the MT will initiate the MO handoff at time k if the averaged received signal strength is less or equal to a predefined MO threshold, MOT_{WLAN}, and the estimated lifetime is less than or equal to the handoff delay threshold, T_{HO}. The MOT_{WLAN} is usually chosen to be a few dB above the wireless interface sensitivity. T_{HO} can be set to the expected handoff delay between the two access technologies. This delay includes several signaling delay components such as discovery delay, authentication delay, and registration delay.

In general, a larger averaging window size results in better estimation but also larger delay in handoff performance [7]. Hence, using variable window sizes that adapt to the MT mobility can improve handoff performance. For example, $W_{average}$ and W_{slope} can be determined by $W_{average} = max\left(10, \left\lfloor \frac{D_{average}}{VT_{sampling}} \right\rfloor\right)$ and $W_{slope} = 2 * max\left(50, \left\lfloor \frac{D_{slope}}{VT_{sampling}} \right\rfloor\right)$, where $D_{average}$ and D_{slope} represent the averaging and slope distance windows respectively, $T_{sampling}$ is the sampling interval used in sampling the RSS values, $\lfloor \cdot \rfloor$ represents the greatest lower integer function, and V is the MT velocity away from the AP, which can be obtained by many velocity estimators proposed in the past, for example [18].

In the MI decision, a main factor is the availability of the WLAN, which can be determined by the RSS of the WLAN. Other factors including the QoS, specified in terms of the available bandwidth, security, and user preference can also be considered. In this work, we assume a simplified model where the MT performs MI to the WLAN if $\overline{RSS}[k] > MIT_{WLAN}$ and the available bandwidth is greater than the required bandwidth. In our simulation and analysis, we assume that the WLAN is always in good condition, so that the MT always perform an MI after an unnecessary MO.

In the next section, we provide an analytical framework for evaluating the performance of the proposed cross-layer adaptive vertical handoff.

4 Performance Analysis

4.1 Transition Probabilities

The calculation of the transition probabilities is based on recursive computation of the handoff probabilities similar to [19]. In the following analysis, we consider

- $P_W[k]$: Pr{MT is associated with the WLAN at instant k}
- $P_C[k]$: Pr{MT is associated with the 3G network at instant k}

- $P_{W|C}[k]$: Pr{MT associates itself with the WLAN at instant k given that it is associated with the cellular network at instant k-1}
- $P_{C|W}[k]$: Pr{MT associates itself with the 3G network at instant k given that it is associated with the WLAN at instant k-1}

In our model, the MT is assumed to be attached to the WLAN at the beginning; hence $P_W[0] = 1$ and $P_C[0] = 0$. $P_W[k]$ and $P_C[k]$ can be calculated recursively:

$$P_W[k+1] = P_{W|C}[k+1]P_C[k] + \left(1 - P_{C|W}[k+1]\right) P_W[k] \, , \tag{3}$$

$$P_C[k+1] = P_{C|W}[k+1]P_W[k] + \left(1 - P_{W|C}[k+1]\right) P_C[k] \, . \tag{4}$$

The Conditional probabilities $P_{C|W}[k+1]$ and $P_{W|C}[k+1]$ depend on the handoff initiation strategy of the algorithm. For the proposed cross-layer adaptive lifetime based VHO algorithm, $P_{C|W}[k+1]$ is determined by

$$P_{C|W}[k+1] = Pr\left\{\overline{RSS}[k+1] < MOT_{WLAN}, EL[k] < T_{HO} | W[k]\right\} \ , \tag{5}$$

where $W[k]$ represents the event that the MT is associated with the WLAN at time $k-1$. In practice, WLAN's are designed for low mobility users. The lifetime part of the MO condition becomes more significant for low mobility users. Hence the MO condition can be reduced to $EL[k] < T_{HO}$. Consequently, one can determine $P_{C|W}[k+1]$ as follows:

$$\begin{aligned} P_{C|W}[k+1] &= Pr\left\{EL[k+1] < T_{HO} | EL[k] > T_{HO}\right\} \qquad (6) \\ &= Pr\left\{\overline{RSS}[k+1] - T_{HO}S[k+1] < \gamma | \overline{RSS}[k] - T_{HO}S[k] > \gamma\right\} \ . \end{aligned}$$

Let $Z[k] = \overline{RSS}[k] - T_{HO} * S[k]$. Then we have

$$P_{C|W}[k+1] = \frac{Pr\{Z[k+1] < \gamma, Z[k] > \gamma\}}{Pr\{Z[k] > \gamma\}} \, . \tag{7}$$

Clearly, since $RSS[k]$ is a Gaussian process, the processes $\overline{RSS}[k]$ and $S[k]$ are Gaussian, and hence $Z[k]$ are Gaussian too. Let its mean be $\mu_Z[k]$ and standard deviation be $\sigma_Z[k]$. It can be shown that [20]

$$\mu_Z[k] = \mu_{\overline{RSS}}[k] - T_{HO}\, \mu_S[k] \ , \tag{8}$$

where $\mu_{\overline{RSS}}[k] = \mu_{RSS}[k] + \frac{1}{W_{average}} \sum_{i=0}^{W_{average}-1} 10nlog(1 - \frac{iVT_{sampling}}{d[k]})$, and $\mu_S[k] = \frac{E\{M_1[k]\} - E\{M_2[k]\}}{W_{slope}T_{sampling}}$, and furthermore

$$\sigma_z^2[k] = \sigma_{\overline{RSS}}^2[k] + T_{HO}^2 \sigma_S^2[k] + \frac{4T_{HO}\sigma_{RSS}^2 \sum_{h=0}^{h=W_{average}-1} (W_{average} - |h|)}{W_{slope}^2 T_{sampling} W_{average}^2} \tag{9}$$

where $\sigma_{\overline{RSS}}^2[k] = \frac{\sigma^2}{W_{average}}$ and $\sigma_S^2[k] = \frac{4\sigma^2}{(T_{sampling}W_{slope}^2 W_{average})^2} \times \left[W_{average}W_{slope} + \sum_{h=1}^{W_{average}-1} (W_{average} - |h|)(2W_{slope} - 6\,|h|)\right]$. Additionally,

$Z[k]$ and $Z[k-1]$ are jointly Gaussian with correlation coefficient $\rho_{Z[k],Z[k-1]}$ as derived in the Appendix, which defines their joint PDF $f_{Z[k]Z[k-1]}(z_1, z_2)$ [21].

Then we can compute $P_{C|W}[k+1]$ by

$$P_{C|W}[k+1] = \frac{\int_{-\infty}^{\gamma} \int_{\gamma}^{\infty} f_{Z[k+1]Z[k]}(z_1, z_2)\, dz_1\, dz_2}{Q\left(\frac{\gamma - \mu_{Z[k]}[k]}{\sigma_{Z[k]}[k]}\right)}\,, \tag{10}$$

where Q(x) is the complementary error function. Similarly, $P_{W|C}[k+1]$ can be determined by

$$P_{W|C}[k+1] = \frac{Pr\{\overline{RSS}[k+1] > MIT, \overline{RSS}[k] < MIT\}}{Pr\{\overline{RSS}[k] < MIT\}}\,. \tag{11}$$

where, similar to the (Z[k+1],Z[k]) tuple, the $(\overline{RSS}[k{+}1], \overline{RSS}[k])$ tuple is jointly Gaussian. These transition probabilities are used to calculate the performance metrics as follows.

4.2 Handoff Probabilities and the Number of Handoffs

The number of handoffs has major impact on the signaling traffic, which may overload the network resulting in degradation in the overall performance. The number of handoffs, denoted N_{HO}, is defined as the sum of MO's and MI's between WLAN and 3G network as the MT roams across the network boundary. Hence, it is a random variable that depends on the instantaneous move out/in probabilities, which can be calculated by

$$P_{MO}[k+1] = P_{C|W}[k+1]P_W[k]\,, \quad P_{MI}[k+1] = P_{W|C}[k+1]P_C[k]\,. \tag{12}$$

The MT movement between the two networks can be modeled by a two-state non-homogeneous Markov chain, where each state represents the network with which the MT is associated. The transition probabilities are $P_{MO}[k]$, from WLAN, and $P_{MI}[k]$, from 3G. Hence, by using binary impulse rewards for the handoff transition as shown in [22], we calculate the average accumulated rewards for MO and MI transitions, which are equivalent to the expected number of MO's, N_{MO}, and the expected number MI's, N_{MI}, respectively. Hence, the expected number of handoffs can be calculated by

$$E\{N_{HO}\} = E\{N_{MO}\} + E\{N_{MI}\} = \sum_{k=1}^{k_{max}} \left(P_{MO}[k] + P_{MI}[k]\right)\ . \tag{13}$$

4.3 Available Bandwidth

The available bandwidth to the MT depends on the proportion of time that the MT stays in the WLAN and the 3G network, as well as the state of the WLAN when the MT is connected to the WLAN. To the MT, the WLAN is in one of two states: WLAN Up and WLAN Down. The WLAN Up state represents the event that the WLAN signal received at the MT is above the sensitivity level α.

WLAN Down is the reverse case. Let $p[k]$ be the probability that the WLAN is in the Up state at time k. Clearly

$$p[k] = Pr\{RSS[k] > \alpha\} = Q\left(\tfrac{\alpha - \mu[k]}{\sigma}\right) . \tag{14}$$

Based on the adopted handoff algorithm, the MO distance varies; consequently the captured WLAN Up durations does too. For the rest of the analysis, we are interested in evaluating the system performance during the *transition region*, which is defined as the range of distance between the point when the RSS starts to oscillate around the interface sensitivity and the WLAN edge. The determination of the transition region is equivalent to a long-standing complex level crossing problem that is analytically tractable only for a few simple cases and is usually solved numerically for complex cases. Here, we obtained the transition region starting bound, denoted k_{start}, from rough estimates based on simulation results.

Then, the WLAN efficiency, ζ_{LT}, defined as the percentage of the WLAN up duration over the MT lifetime in the WLAN, can be estimated as

$$\zeta_{LT} = \sum_{k=k_{start}}^{k_{max}} \overline{P_{MO}}[k] \frac{\sum_{h=1}^{k} p[h]}{k} , \tag{15}$$

where $\overline{P_{MO}}$ is a scaled version of P_{MO} to represent a valid PDF within interval $[1, k_{max}]$, and k_{max} represents the time index at which the MT reaches the WLAN edge and is determined by the planed coverage area.

Hence, the MT available bandwidth, BW_{Av}, assuming R_W and R_C as the effective data rates in WLAN and cellular networks respectively, can be computed as

$$BW_{Av} = \frac{\zeta_{LT} R_W(\overline{k_{MO}} - k_{start}) + R_C(k_{max} - \overline{k_{MO}})}{(k_{max} - k_{start})} , \tag{16}$$

where $\overline{k_{MO}}$ denotes the average time to MO.

4.4 Packet Delay

In addition to the MT available bandwidth, RSS degradation in the transition region impacts on the head of line (HoL) packet delay probability. To study this, we assume a threshold, θ_D for packet delay in the current hop as a part of the end-to-end delay budget for the real-time application packet from the source to the destination. A packet is considered *excessively delayed* if its HoL delay exceeds θ_D[1]. Consequently, the average packet delay probability, D, can be estimated as

$$D = \frac{\sum_{k=k_{start}}^{\overline{k_{MO}}} P_D[k]}{(\overline{k_{MO}} - k_{start} + 1)} , \tag{17}$$

[1] Note that this does not necessarily mean that the packet is lost.

where $P_D[k]$ represent the probability that a packet will be excessively delayed, which is equal to the probability of WLAN Down runs whose duration is equal to the delay threshold. Here we have performed an approximation by using $\overline{k_{MO}}$, instead of using k_{MO} and then applying conditional expectation. As shown in the next section, this approximation produces accurate results over a wide range of system parameters.

5 Numerical Results and Simulation

5.1 Simulation Model

In addition to analysis, we have simulated VHO using MATLAB. The simulation model assumes a MT moving away from the WLAN access point with constant speed V. Table 1 shows the values of the simulation parameters. The system parameters are used as in [15]. These parameters result in a WLAN coverage of approximately 100 meters in radius. In the following, each point in the simulation results represent the average of 100 runs.

Table 1. Simulation parameters values

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
P_T	100 mWatt	n	3.3	σ	7 dB	S	28.7 dB
$D_{average}$	0.5 m	D_{slope}	5 m	α	-90 dBm	$T_{sampling}$	0.01 sec
MOT_{WLAN}	-85 dBm	MIT_{WLAN}	-80 dBm	$T_{handoff}$	1 sec	$\frac{R_W}{R_C}$	$\frac{25\text{Mbps}}{2.4\text{Mbps}}$

5.2 Performance Comparison

We compare the performance of adaptive VHO with traditional hysteresis VHO, which is used in [2]. In hysteresis based algorithms, there are two different thresholds MIT_{WLAN} and MOT_{WLAN} for the MI and MO respectively. The MT performs a MI if the $\overline{RSS}[k]$ is larger than MIT_{WLAN} and performs a MO if $\overline{RSS}[k]$ is smaller than the predefined MOT_{WLAN}. Usually, MIT_{WLAN} is chosen larger than MOT_{WLAN} to decrease the number of unnecessary handoffs known as ping-pong effect.

Due to page limitation, we will simply state that the introduction of the adaptive lifetime approach to the traditional hysteresis VHO algorithm results in significant decrease of the number of unnecessary handoffs and significant improvement on the available bandwidth [20]. Clearly, from a pure bandwidth point of view, it is preferable for the MT to perform MO handoff near the WLAN edge, even though the RSS can temporarily go below the MT sensitivity level in the transition region. However, a drawback of increasing the lifetime of the MT within the WLAN is increasing the packet delay resulting from channel degradation. As shown in [20], the packet delay probability using the adaptive approach can be much more than that when the traditional hysteresis algorithm is used. This may be critical if the MT is running real-time application. However, by the proper tuning of the ASST as shown in the next subsection, this effect can be adapted to the active real-time application requirements in the MT.

5.3 Application Signal Strength Threshold Adaptation

Figures 1, 2, and 3 illustrate the effect of the ASST on the number of handoffs, available bandwidth, and packet delay probability. They show that the number of handoffs decreases when the ASST is reduced, since reducing the ASST allows the MT to remain in the WLAN for a longer duration. For the same reason, the available bandwidth to the MT increases when the ASST is reduced. However, at the same time, the packet delay probability is increased, since signal outage is more severe near the edge of the WLAN. Hence, there is a clear trade off among the handoff signaling load, available bandwidth, and packet delay.

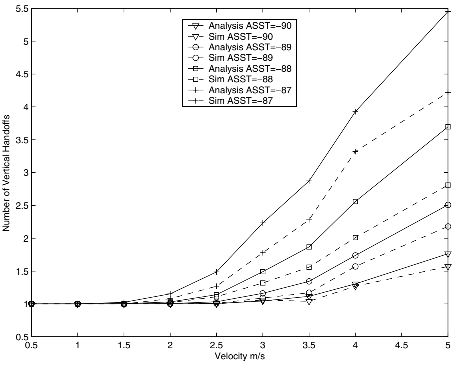


Fig. 1. Number of Handoffs vs. ASST

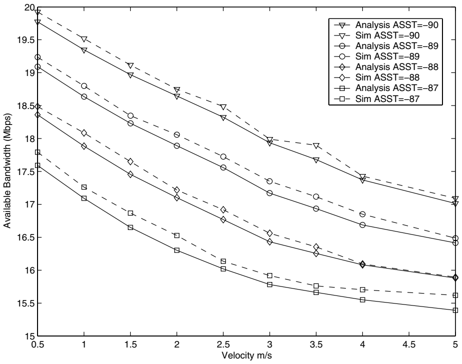


Fig. 2. Available Bandwidth vs. ASST

Clearly, the ASST should not depend on the application QoS alone. Rather, it can be optimally tuned based on the various conflicting criteria of VHO. Likewise, the optimal VHO decision can be made adaptive to the RSS variation, network delay characteristics, and application QoS demands, through a properly chosen ASST value. The propose analytical framework provides a means to carry out this optimization.

As an example, a possible cost function to aggregate the multiple VHO criteria may be $C_{total} = \frac{c_H E\{N_{OH}\} + c_D D}{BW_{av}}$, where c_H represents the signaling cost per handoff, c_D represents the penalty factor for packet delay, and C_{total} is normalized to cost per Mbps of data bandwidth[2].

Figure 4 plots C_{total} over different ASST values, for $V = 2$, $c_H = 100$, and $c_D = 10000$, where each curve represents a delay threshold value of 40 ms,

[2] We emphasize here that this is only one of many possible cost functions, whose suitability depends on practical application goals and system constraints.

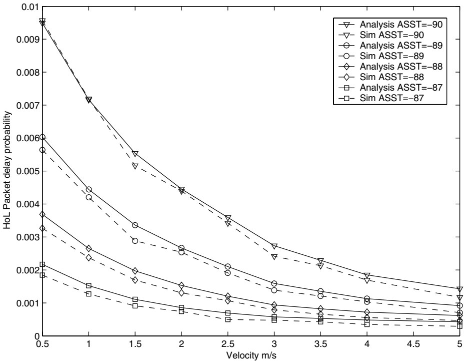


Fig. 3. HoL Packet Delay Rate vs. ASST (θ_D= 30ms)

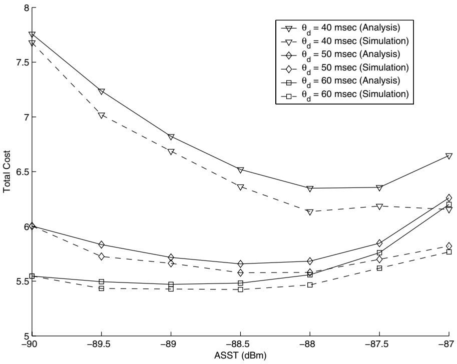


Fig. 4. Total Cost vs. ASST ($V = 2$)

50 ms, and 60 ms, respectively. Clearly, the optimal ASST increases as the delay threshold decreases. In particular, when $\theta_D = 40ms$, an optimal ASST of -88 dBm strikes the optimal balance to minimize the total cost, but when $\theta_D = 60ms$, the optimal ASST is -89 dBm. To implement this in practice, a lookup table can be built based on the proposed numerical analysis results.

6 Conclusions

We have presented a cross-layer adaptive VHO approach that takes into account the wireless channel variation, network layer latency, and application QoS demands. We have proposed an analytical framework to evaluate the performance of VHO based on multiple criteria. The proposed application signal threshold adaptation provides a means for flexible system design. Given a predefined priority policy, it can be used to optimize the tradeoff between handoff signalling, available bandwidth, and packet delay. Since the ASST can be optimally tuned for any access network based on practical system characteristics and requirements, it may have a significant role in future generation wireless networks where access technologies with vastly differing characteristics are expected to seamlessly co-exist and efficiently inter-operate.

References

1. Berezdivin, R., Breinig, R., Topp, R.: Next-generation wireless communications concepts and technologies. IEEE Communications Magazine (2002)
2. Buddhikot, M.M., Chandranmenon, G., Han, S., Lee, Y.W., Miller, S., Salgarelli, L.: Integration of 802.11 and third generation wireless data networks. In: Proc. of IEEE INFOCOM, San Francisco, US (2003) 503–512

3. Salkintzis, A.K., Fors, C., Pazhyannur, R.: WLAN-GPRS integration for next-generation mobile data networks. IEEE Wireless Commun. **9** (2002) 112–124
4. Akyildiz, I.F., McNair, J., Ho, J., Uzunalioglu, H., Wang, W.: Mobility management in current and future communications networks. IEEE Network **12** (1998) 39–49
5. Liang, B., Haas, Z.J.: Predictive distance-based mobility management for multidimensional pcs networks. IEEE/ACM Transactions on Networking **11** (2003) 718–732
6. Makela, J., Ylianttila, M., Pahlavan, K.: Handoff decision in multi-service networks. In: Proc. of 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'00). Volume 1., London, UK (2000) 655–659
7. Pollini, G.P.: Trends in handover design. IEEE Commun. Mag. **34** (1996) 82–90
8. Wang, H., Katz, R.H., Giese, J.: Policy-enabled handoffs across heterogeneous wireless networks. In: Proc. of the Second IEEE Workshop on Mobile Computer Systems and Applications, New Orleans, Louisiana (1999) 51
9. Hatami, A., Krishnamurthy, P., Pahlavan, K., Ylianttila, M., Makela, J., Pichna, R.: Analytical framework for handoff in non-homogeneous mobile data networks. In: Proc. of (PIMRC'99), Osaka, Japan, (1999) 760–764
10. Ylianttila, M., Pande, M., Makela, J., Mahonen, P.: Optimization scheme for mobile users performing vertical hand-offs between IEEE 802.11 and GPRS/EDGE networks. In: Proc. of IEEE Global Telecommunications Conference GLOECOM'01. Volume 6., San Antonio, Texas, USA (2001) 3439–3443
11. Ylianttila, M., Makela, J., Mahonen, P.: Supporting resource allocation with vertical handoffs in multiple radio network environment. In: Proc. of IEEE International Symposium on Personal, Indoor and Mobile Radio Com-munications (PIMRC'02), Lisbon, Portugal (2002) 64–68
12. Zhu, F., McNair, J.: Optimizations for vertical handoff decision algorithms. In: in Proc. of IEEE Wireless Communications and Networking Conference (WCNC). Volume 2. (2004) 867–872
13. Ylianttila, M., Pichna, R., Vallstram, J., Makela, J., Zahedi, A., Krishnamurthy, P., Pahlavan, K.: Handoff procedure for heterogeneous wireless networks. In: Proc. of IEEE Global Telecommunications Conference (GLOBECOM'99). Volume 5. (1999) 2783–2787
14. Pahlavan, K., Krishnamurthy, P., Hatami, A., Ylianttila, M., Makela, J.P., Pichna, R., Vallstron, J.: Handoff in hybrid mobile data networks. IEEE Commun. Mag. **7** (2000) 34–47
15. Majlesi, A., Khalaj, B.H.: An adaptive fuzzy logic based handoff algorithm for hybrid networks. In: Proc. of 6th International Conference on Signal Processing. Volume 2. (2002) 1223–1228
16. Buddhikot, M.M., Chandranmenon, G., Han, S., Lee, Y., Miller, S., Salgarelli, L.: Design and implementation of a WLAN/CDMA2000 interworking architecture. IEEE Commun. Mag. **41** (2003) 90–100
17. Rappaport, T.S.: Wireless Comm.: Principles and Practice. Prentice Hall (1999)
18. Tepedelenlioglu, C., Giannakis, G.: On velocity estimation and correlation properties of narrow-band mobile communication channels. IEEE Trans. on Vehicular Technology **50** (2001) 1039– 1052
19. Zhang, N., Holtzman, J.M.: Analysis of handoff algorithms using both absolute and relative measurements. IEEE Transactions on Vehicular Technology **45** (1996) 174– 179

20. Zahran, A.H., Liang, B.: ALIVE-HO: Adaptive lifetime vertical handoff for heterogeneous wireless networks. Technical report, University of Toronto (2004)
21. Papoulis, A., Pillai, S.: Probability, Random Variables and Stochastic Processes. 4th edn. McGraw-Hill (2002)
22. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Queuing networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. 2nd edn. Wiley (1998)

Justice: Flexible and Enforceable Per-Source Bandwidth Allocation

Jakob Eriksson, Michalis Faloutsos, and Srikanth Krishnamurthy

University of California, Riverside
{jeriksson, michalis, krish}@cs.ucr.edu

Abstract. The main goal of this work is to provide bandwidth allocation that is robust against the behavior of greedy or malicious users. The traditional solution, Fair Queueing, allocates capacity per source-destination pair in accordance with the max-min fairness criterion. While Fair Queueing, defined as above, has been successful and popular to a large extent, it does not prevent greedy or malicious users from getting unfair shares of capacity. In particular, it is vulnerable to end-points simply establishing *multiple parallel connections* to increase their allocated capacity.

In order to overcome this limitation, we propose Justice, which allows for robust, yet flexible bandwidth allocation in the Internet. Justice employs *weighted* per source bandwidth allocation to accommodate traffic sources with varying bandwidth requirements. We describe an efficient and scalable mechanism for determining, for each source s, the weight $\phi_k^{(s)}$ at any given link k. We demonstrate through analysis and simulation that Justice is flexible, efficient, scalable, and robust to all identified attacks related to bandwidth allocation.

Keywords: Bandwidth Allocation, Per-Source Weighted Fairness.

1 Introduction

Currently greed and mischief pay off in terms of performance in the Internet. Through a number of reasonably simple tricks, greedy or malicious users can improve their own throughput at the cost of their well-behaved counterparts. The primary goal of this work is to make the bandwidth allocation of each user independent of the potentially malicious behavior of other users of the network.

The problem of isolating user behavior has been addressed before. In particular, Fair Queuing was introduced in RFC 890 [1], where Nagle advocated per-source max-min bandwidth allocation. Shortly thereafter, Demers, Keshav and Shenker, [2] brought up the fact that some sources may deserve a higher bandwidth allocation than others, making a per-source max-min allocation impractical. Their proposed solution was to instead allocate bandwidth per source-destination pair. Unfortunately, this solution makes Fair Queuing vulnerable to users sending traffic to multiple destinations to maximize their total throughput. An ideal bandwidth allocation scheme needs to be both flexible enough to

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1206–1218, 2005.

accommodate a wide variety of traffic sources, and enforceable so that a well-behaved source can retain its allocation in despite the actions of malicious or greedy sources.

We propose Justice[1], a novel approach to bandwidth allocation. Justice provides a guaranteed minimum allocation to each source, at each link, independent of the behavior of other users. With the protection of well-behaved users as its first concern, Justice employs weighted per-source fairness at each link, and provides a mechanism for efficiently determining a unique per-source weight $\phi_k^{(s)}$ of each source s, at each link k in the network.

Justice exhibits the following properties. First, it is **robust**: through the careful computation of per-source weights, Justice ensures that greedy or malicious sources cannot, through manipulation of the weight calculation process, affect the bandwidth allocation to their advantage. Second, it is **enforceable**: the use of per-source fairness at each link ensures that the actual bandwidth achieved corresponds closely to the allocation assigned, independent of the behavior of competing traffic sources. Third, it is **flexible**: Justice computes a unique per-source weight for each source at each link. This provides administrators with the flexibility to allocate extra bandwidth to sources that deserve it. Fourth it is **scalable**: Administration is limited to local per-link settings, as opposed to global per-source weights. Moreover, a router needs only keep track of the weights of sources that are actively using a link incident on it, the weights of inactive sources are ignored. These properties have previously been achieved in isolation, but to the best of our knowledge, Justice is the first to provide all of them together.

2 Background and Motivation

Much research has gone into improving bandwidth allocation in the Internet. It is well established that FIFO-Droptail queuing, where packets are served in arrival order, and arriving packets are dropped if they find the queue full on arrival, does not provide an adequate level of protection. Aside from the congestion control algorithm used in TCP [3], most of the work has been done at the router level.

Queue Management algorithms such as RED [4] with variations [5] [6] [7] and CHOKe [8] attempt to notify and punish overly aggressive flows. These techniques rely on having conforming TCP clients at the endpoints. Moreover, other link-based schemes, like Weighted Round-Robin queuing, or proportional random dropping suffer from similar problems, in particular in conjunction with TCP congestion control, where aggressive flows can often cause TCP to back off indefinitely [9].

An approach toward isolating user behavior which has received much attention is that of Fair Queuing [10, 11, 1, 2, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. In general, the goal of Fair Queuing is to provide max-min fairness between

[1] The name Justice has been used by us in previously submitted work. Please note that although the high-level goal is similar, the proposed scheme is significantly different.

flows, where a flow can be defined in several ways, and with the implicit assumption that all flows have equal rights to network resources. In the original RFC [1], Nagle proposed to provide fairness per source. However, in [2], the authors find that per-source fairness is impractical, as some hosts, such as file servers, *deserve* a larger allocation of bandwidth. They suggest that this problem could be solved by providing routers with knowledge about the *deserved* allocation of each source, an approach commonly referred to as *weighted per-source fairness*. However, they do not provide a mechanism to determine and distribute such knowledge. Instead, they propose to provide max-min fairness between *conversations*, or source-destination pairs. However, quoting [2]: *"Allocation per source-destination pair allows a malicious source to consume an unlimited amount of bandwidth by sending many packets all to different destinations."* This vulnerability of per-conversation bandwidth allocation has yet to be successfully addressed in the literature.

Link Sharing is a related technique, which focuses on sharing bandwidth between incoming links or different classes of traffic. Examples of this are Class Based Queuing [22], and Weighted Fair Queuing [23]. These are also referred to as Differentiated Services policies [24]. In general, such approaches provide protection between traffic classes or between incoming links, but are powerless to provide protection between sources in the same class, or the same link, respectively.

We conclude that current approaches are not sufficient to protect well-behaved users from their greedy or malicious counterparts. Due to the inherent vulnerability of per-conversation allocation, we propose to use weighted per-source allocation. To accommodate sources that deserve a higher allocation than others, we provide an efficient mechanism for determining the unique weight $\phi_k^{(s)}$ of source s at link k, thus solving the original problem with per-source Fair Queuing.

Like other schemes based on Fair Queuing, we assume that the source address in the packet header is accurate. Under certain circumstances, an attacker could potentially falsify the source address of his packets to improve his allocation. A countermeasure against source address falsification is described in [25].

3 Defining Justice

In this section we will describe Justice as a concept, leaving implementation details for later. Justice constitutes a new class of source-based bandwidth allocation policies, where the primary objective is to protect well-behaved sources from their malicious or greedy counterparts. At a high level, Justice combines a network-wide method for determining the appropriate per-source weight $\phi_k^{(s)}$ of each source s at any link k, with weighted per-source fairness[2] locally on each router.

[2] Weighted per-source fairness is a variation of per-flow fairness where all traffic from a given source is treated as a single flow, and where each source may have a unique configured weight, indicating its *deserved* allocation on a given link.

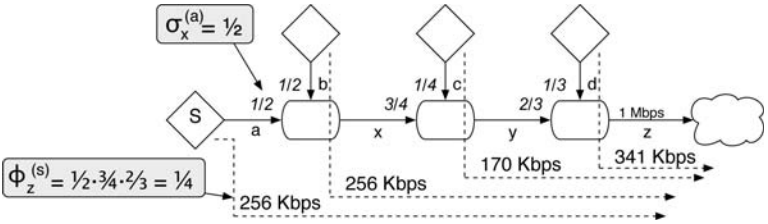


Fig. 1. Bandwidth allocation with Justice. The allocation of source s on link z, $\phi_z^{(s)}$, is determined by the per-link σ settings on the routers between s and z

Until now, weighted per-source bandwidth allocation has not been practical, because there has been no adequate mechanism for determining the appropriate weight of each source at each link, or for the distribution of this knowledge.

With Justice, we propose an efficient and scalable method for determining the weight $\phi_k^{(s)}$ of each source s at each link k, without the storage and administration overhead of configuring this weight at each router. The per-source weight is computed based on a set of configured per-link weights $\sigma_x^{(y)}$ along the path from the source to the link in question. Figure 1 shows an example of how bandwidth is allocated based on configured per-link weights. Here, link a has a weight $\sigma_x^{(a)} = \frac{1}{2}$ on the outgoing link x. Similar per-link weights are configured throughout the network. **Justice defines the per-source weight $\phi_k^{(s)}$ of source s at link k, as the product of the per-link weights along the path from s to k.** In this case, $\phi_z^{(s)} = \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{2}{3} = \frac{1}{4}$.

Note that the configured weight on each link could be changed arbitrarily, which would then result in a different bandwidth allocation. In general, Justice is very flexible with respect to the final allocation. For example, a naive approach to link weight assignment would be to assign the same weight to all links. This would result in an allocation where many users sharing a high capacity link are disadvantaged, as compared to a small number of users sharing a low capacity link. A more practical, but still naive, approach might be to assign weights according to link capacity. Finally, Justice also makes more advanced traffic engineering approaches possible, one of which we will describe in Section 5.

More formally, for each pair of links x,y incident on a router, a constant weight $\sigma_y^{(x)}$ is *configured.* We define $\sigma_y^{(x)}$ to be the fraction of the capacity of link y, that is to be the guaranteed minimum allocation for packets that arrive through link x. $\sigma_y^{(x)}$ is a value between 0 and 1, and for every outgoing link y

$$\sum_{x \neq y} \sigma_y^{(x)} = 1. \tag{1}$$

Given a path consisting of links $0, 1, \ldots, k$, we define the minimum weight $\phi_k^{(s)}$ of source s at link k, to be

$$\phi_k^{(s)} = \prod_{i=0}^{k-1} \sigma_{i+1}^{(i)} \tag{2}$$

Note that $\sigma_y^{(x)}$ describes the minimum per-link weight. Justice is work-conserving, so should packets arriving from an incoming link use less bandwidth than what that link's weight entitles it to, the surplus bandwidth on the outgoing link is distributed between the other incoming links in proportion to their respective weights, as discussed in the following section.

4 Enforcing Justice

In this section we will discuss how Justice can be practically implemented in a network. Enforcing Justice consists mainly of two challenges. First, an efficient mechanism for calculating the $\phi_k^{(s)}$ of each active source s on a link k is required. Second, it is necessary to ensure that each source actually gets its deserved $\phi_k^{(s)}$ share of the capacity of the link. The second part can be accomplished by using per-source Weighted Fair Queuing.

Described briefly, per-source Weighted Fair Queuing (WFQ) keeps a separate queue for each active traffic source, and sends packets from each queue in proportion to their respective weights. WFQ is well understood [23, 11], and has been shown to fairly distribute bandwidth between flows according to preconfigured weights. WFQ requires per-source state. In most routers, this is not a concern. However, in core routers, keeping per-source state may result in higher hardware costs. If per-source state in core routers is a concern, Justice can be implemented in a manner analogous to core-stateless Fair Queuing [21], as we report in [9]. This technique removes the need for per-source state in core routers, but is not described here due to space constraints. We will now focus on the efficient calculation and distribution of per-source weights.

We will now demonstrate how simple per-packet calculations can be used to compute the current per-source weight at any given link. Let us define $in_r(p)$ and $out_r(p)$ to mean the incoming and outgoing link of a packet p, at router r on the path from source to destination, and let $src(p)$ be the source of the packet. For every incoming packet p, a router needs an efficient and secure method to determine the share $\phi_{out(p)}^{(src(p))}$, that the packet's source $src(p)$ deserves on the outbound link, $out(p)$. From Eq. 2, we know that the weight of source $src(p)$ on the r:th outbound link $out_r(p)$ is

$$\phi_{out_r(p)}^{(src(p))} = \prod_{i=0}^{r} \sigma_{out_i(p)}^{(in_i(p))} . \tag{3}$$

It is easy to see that this can also be written as the recursion

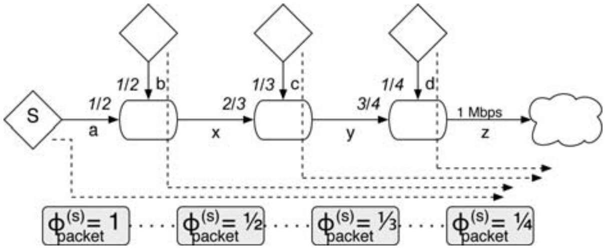

1/2
b
1/3
c
1/4
d
S
1/2
a
2/3
x
3/4
y
1 Mbps
z
$\phi^{(s)}_{packet}$ = 1
$\phi^{(s)}_{packet}$ = ½
$\phi^{(s)}_{packet}$ = ⅓
$\phi^{(s)}_{packet}$ = ¼

Fig. 2. Iterative calculation of weights along a path from left to right. $\phi^{(s)}_{packet}$ indicates the weight of source s on the current outbound link, stored in the packet header

$$\phi^{(src(p))}_{out_r(p)} = \phi^{(src(p))}_{out_{r-1}(p)} \cdot \sigma^{(in_r(p))}_{out_r(p)} . \tag{4}$$

This observation suggests that the sequence of per-source weights along a path from source to destination can be computed iteratively, one hop at a time, as shown in Figure 2. In order to do this, we store weight of the source at the current link in the packet header. Let us call this value ϕ_{packet} Upon receiving a packet, router r can compute $\phi^{(src(p))}_{out_r(p)}$ by simply multiplying ϕ_{packet} by the incoming link's *configured* share of the next hop, $\sigma^{(in_r(p))}_{out_r(p)}$.

However, since the weight calculation now depends on receiving accurate ϕ_{packet} values from upstream routers, it is vulnerable to manipulation. A router could conceivably set the weights in its outgoing packets to artificially high values.This could inflate the downstream allocation of traffic passing through that router, giving those sources an unfair advantage. To address this vulnerability, we require that incoming ϕ_{packet} weights of any given link be normalized, so that $\sum_{s \in S} \phi^{(s)}_k = 1$. We call this **source-weight normalization**.

$$\phi^{(src(p))}_{out_r(p)} := \frac{\phi^{(src(p))}_{out_r(p)}}{\sum_{s \in S} \phi^{(s)}_{out_r(p)}} \tag{5}$$

Source-weight normalization has two major benefits. First, it effectively guards against manipulation of the ϕ_{packet} weights reported in packet headers, since whatever weights come in, their sum will always be equal to one after the normalization step, and thus will not adversely affect the allocation of sources using other links. Second, equation 5 scales the weight of active sources proportionally so that they add up to one. If some sources are not using their allocated share of the link, source-weight normalization will scale the weights of all active sources proportionally to fill any unallocated portion.

Like in core-stateless Fair Queuing [21], we assume that a value (ϕ_{packet}) can be stored in the packet header. There are several locations where the value can be stored. The 8-bit Type-of-Service field in the IPv4 header is sufficient to approximate Justice, as discussed in [9]. Other possibilities include the ToS and Flow ID fields in IPv6. Finally, ϕ_{packet} can be communicated at the link-layer.

This may be the best solution, as it leaves the IP header unmodified throughout the network.

5 Analysis

Given the mechanism provided for calculating per-source weights, ϕ, it seems natural to ask what is the proper configuration of link weights, σ. In this section, we will use analysis to study the effect a given set of σ values and other factors will have on the throughput of a flow. We will also showcase how Justice can be used for traffic engineering. In the scenario we will discuss, a set of traffic sources distributed across a large network need to share a single bottleneck link. We will describe a general method for how to allocate link weights to achieve any desired allocation on the bottleneck link.

Previous sections have discussed only the computation and distribution of per-source weights. While these accurately describe the minimum weight of a source at a *given* link, they say nothing about the actual throughput that can be expected during end-to-end communication. Let us define C_k to be the bandwidth capacity of link k, and $\gamma_k^{(s)}$ to be the minimum achieved throughput of source s on the path to, and including, link k. Where the identity of the source is unambiguous, we write simply γ_k.

Given a source s, a path consisting of links $0, 1, \ldots, i$, and an initial sending rate of R, it is clear that on the first link $\gamma_0^{(s)} = min(R, \phi_0^{(s)} C_0)$, since $\phi_0^{(s)} C_0$ represents the minimum guaranteed bandwidth allocation on link 0. In general, however, γ_k also depends on the achieved bandwidth through the previous link, γ_{k-1}, or

$$\gamma_k = min(R, \gamma_{k-1}, \phi_k C_k) \tag{6}$$

Theorem 1. *$\gamma_k^{(s)}$ is the guaranteed minimum throughput that source s can expect to see across the sequence of links $0, 1, \ldots, k$.*

Proof. We will proceed by induction. Assume that the first link, link 0, is exclusively used by the source. [3] In the base case, γ_1, there is a single source using link 0, we have $\phi_1 = \sigma_1^{(0)}$. Weighted Fair Queuing (WFQ) provably distributes bandwidth according to the weights provided, so $\gamma_1 = min(R, \phi_1 C_1)$. For the inductive step, assume that γ_{k-1} holds. Accordingly, the input link $k-1$ serves a flow from source s with rate γ_{k-1}. Again, WFQ will serve the flow according to the weight given, ϕ_k in this case. If $\gamma_{k-1} \leq \phi_k C_k$, then WFQ will allocate s γ_{k-1} of bandwidth. Else, it will allocate $\phi_k C_k$, which completes the induction. QED

In most cases, the actual throughput will be considerably higher than the guaranteed minimum throughput. This is because in general, not all sources will be

[3] This assumption can be relaxed to include switched LANs. For shared medium physical layers, the hosts using it are considered one entity and bandwidth allocation has to be resolved at the MAC layer.

sending their entire σ_i of traffic across every link i. A characterization of the actual throughput received cannot be presented here, due to space constraints.

5.1 Traffic Engineering with Justice

In this section, we will showcase how Justice can be used to aid network traffic engineering. In the example below, we will discuss a scenario within a single AS, but Justice can be used equally well for inter-AS traffic engineering.

So far, we have assumed that a network administrator will provide each router with the necessary per-link weights. The per-link configuration regime provides a simple, but powerful tool for specifying the relative importance of *nearby* traffic sources. However, it may well be the case that the network is large, and that the relative importance of *distant* traffic sources needs to be addressed as well.

As a typical example, an Internet Service Provider (ISP) may have multiple access points, serving customers of multiple service classes, all connected to the same back-bone network. If possible, the ISP would like to ensure that all customers in the same service class receive approximately the same level of service, and that the various service classes exhibit a clear difference in performance. We will now describe a method for assigning per-link weights that provably achieves these goals.

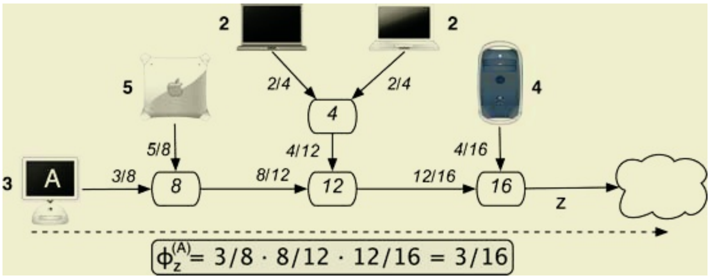


Fig. 3. Assignment of σ values based on prior entitlement information, shown next to each source. The allocation of any source at the bottleneck link will be directly proportional to its entitlement

Let us define *entitlement* η_s, to be an absolute number indicating the amount of the total network resources that source s is entitled to. We can think of η_s as an abstract expected "service level" of a given source. In Figure 3, every traffic source has been marked with an integer entitlement.

For this example, let us assume that all sources have a single bottleneck link, for example the link to the rest of the Internet. Assuming single-path routing, we can form a tree of routing paths, rooted at the gateway. Let us define the *aggregate entitlement* η_r of a router r in the tree to be

$$\eta_r = \sum_{i \in C} \eta_i \tag{7}$$

where C is the set of children of r. In Figure 3, the aggregate entitlement of each router is indicated. The optimal value of $\sigma_y^{(x)}$ can then be computed as

$$\sigma_y^{(x)} = \frac{\eta_n}{\eta_p}, \tag{8}$$

where x is the link between node n and its parent p, and y is the link leading from p toward the root. Figure 3 shows the σ according to Eq. 8 next to each link, and a sample ϕ_z for the traffic source marked "A". Note that the allocation $\phi_z^{(A)}$ equals 3/16, which is the entitlement of source "A", divided by the sum of entitlements in the entire network.

Theorem 2. *If aggregate entitlements and link shares are configured in accordance with Equations 7 and 8, each source s will receive a minimum guaranteed allocation at the bottleneck link b,*

$$\phi_b^{(s)} = \frac{\eta_s}{\sum_{i \in S} \eta_i}, \tag{9}$$

where S is the set of all traffic sources.

Proof. We will again use induction. Consider the base case, where N children are connected to a router r, which in turn is directly connected to the bottleneck link, b. For the base case, the set C in Eq. 7 contains all traffic sources, i.e., all traffic sources are children of r, or $C = S$. Thus, if we substitute Eq. 7 into the denominator of Eq. 8, we get

$$\sigma_b^{(s)} = \frac{\eta_s}{\sum_{i \in S} \eta_i}.$$

It remains only to compute $\phi_b^{(s)}$ from source s to link b which for the base case, according to Eq. 4, equals $1 \cdot \sigma_b^{(s)}$.

In the inductive step, assume that Eq. 9 holds for any tree of depth $\leq d$. We will now show that it holds for any tree T of depth $d+1$. Any tree of depth $d+1$ can be described as a root node r with j trees of depth $\leq d$ as its children. Let us consider a source s which belongs to one of the children of r, the subtree T_s.

We are interested in finding $\phi_b^{(s)}$, where b is the bottleneck link connected to the root node r. According to Eq. 4, this can be written as

$$\phi_b^{(s)} = \phi_{b'}^{(s)} \sigma_b^{(b')}$$

where b' is the link leading from T_s to the root node r. We know by our inductive assumption that

$$\phi_{b'}^{(s)} = \frac{\eta_s}{\sum_{i \in S_{T_s}} \eta_i},$$

where S_{T_s} is the set of all sources in T_s. Moreover, Eqs. 7 and 8 tell us that

$$\sigma_b^{(b')} = \frac{\eta_{T_s}}{\eta_T} = \frac{\sum_{i \in S_{T_s}} \eta_i}{\sum_{i \in S} \eta_i},$$

where S is the set of all sources in T. Finally, putting it all together, we have

$$\phi_b^{(s)} = \phi_{b'}^{(s)}\sigma_b^{(b')} = \frac{\eta_s}{\sum_{i\in S_{T_s}} \eta_i} \cdot \frac{\sum_{i\in S_{T_s}} \eta_i}{\sum_{i\in S} \eta_i} = \frac{\eta_s}{\sum_{i\in S} \eta_i}. \qquad QED$$

Although space does not permit a full discussion, we can offer an intuition on how to use the same framework to compute σ per-link weights from entitlement knowledge when there are more than one bottleneck links in the network. The idea is to form a separate tree for each bottleneck link, and compute the aggregate entitlements separately for each tree. Next, add up the aggregate entitlements out of each tree, for every router, to form the final aggregate entitlement values. Finally compute the σ values according to Eq. 8.

6 Simulation Results

In this section, we present a set of simulations that validate our claims about Justice as a bandwidth allocation scheme. Since the goal of Justice is not the same as that of, for example, per-conversation Fair Queuing, it is not possible to do a quantitative performance comparison. Instead, the purpose of the simulations is to show the behavior of Justice under realistic conditions, and to contrast and compare this with the behavior of previous queuing mechanisms.

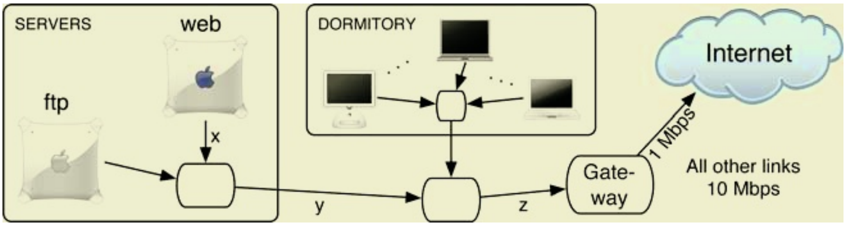
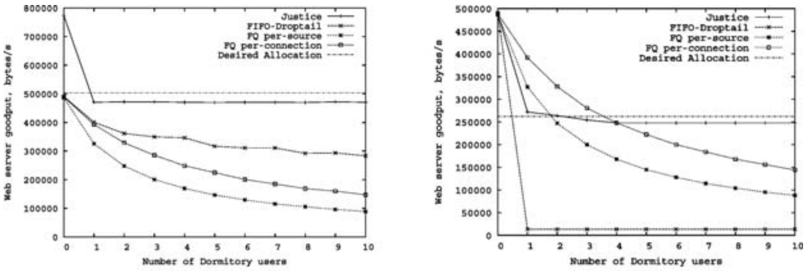


Fig. 4. University campus simulation topology. All intra-campus links are 10 Mbps, but the external link to the Internet is only 1 Mbps

We compare Justice with three other schemes, and the expected minimum allocation according to the analysis in Section 5. In all cases, we use the same queuing policy on all outgoing links on all routers. The current state of the Internet is best represented by the FIFO-Droptail policy, where packets are served in arrival order, and packets are dropped if they find the queue full on arrival. With per-source Fair Queuing, the router keeps a separate output queue per source, and thus ensures that all sources get the same allocation on the outgoing link. Per-flow Fair Queuing takes this one step further, and keeps a separate queue for each established connection, or source-destination pair. In our simulation setup, we assume that each established connection has a unique



(a) Scenario 1 (TCP)

(b) Scenario 2 (UDP)

Fig. 5. Web server goodput under competition from dorm users. Only Justice provides the web server with its deserved bandwidth when there are many competing flows

remote end-point, so there is no distinction between connections and source-destination pairs.

We used the ns-2.27 simulator together with the built in implementation of DropTail queuing, and the CMU implementation of Worst-case Fair Weighted Fair Queuing (WF^{2}Q). We used the WF^{2}Q implementation to simulate per-source Fair Queuing, per-conversation Fair Queuing as well as Justice. For Justice, the ϕ values of all sources on all links were calculated using Eq. 2, and manually configured using WF^{2}Q settings. Our simulations were run on the topology shown in Figure 4. All intra-campus links are of 10 Mbps capacity, but the link to the Internet has a lower 1 Mbps capacity. The two servers each maintain 10 simultaneous TCP connections with end-points in the external Internet. The simulations stabilize quickly, and so it was sufficient to run each simulation for 10 seconds. All transmitted packets are of the same size, 1000 bytes. All results are averaged over 10 runs with randomized starting times for the connections. Except for the FIFO results, std. dev. was near zero.

In our first simulation scenario, results shown in Fig. 5 (a), there is between 0 and 10 dormitory traffic sources, each of which has 5 TCP connections to the Internet. The University network administrator has decided to allocate a minimum of 60% of the capacity on the bottleneck link to the server group, and ensures this by setting $\sigma_z^{(y)} = 0.6$. In addition, the administrator of the server group has decided to allocate 80% of his total capacity to the web server ($\sigma_y^{(x)} = 0.8$), and only 20% to the file server. Using Eq. 6, the expected share of the web server at the bottleneck link is $\gamma_z^{(web)} = 0.8 \cdot 0.6 \cdot 1Mbps = 0.48Mbps$. As expected, Justice effectively enforces the deserved allocation, in contrast with all other schemes which fail to preserve the allocation of the web server as the number of flows and users on the network increases.

Our second simulation scenario has dormitory users using UDP and sending 500 byte packets at a very high constant rate. In addition, the admins have set $\sigma_z^{(y)} = .5$ and $\sigma_y^{(x)} = .5$ which means the intended minimum allocation for the web server is $\gamma_z^{(web)} = 0.25Mbps$. As expected, the FIFO scenario performs very badly here, as allocation depends completely on the cooperativeness of end-hosts. Justice again enforces the allocation configured through the per-link weights. Note that it also enforces the 50% allocation of the dormitory when there is a small number of users, in this case protecting dorm users against the many flows of the web and ftp servers. Although the two Fair Queuing policies show a performance similar to what they showed in the TCP scenario, they still give the web server a progressively smaller share as the number of users in the dormitory increases.

7 Conclusion

In this paper we have presented Justice, a new take on bandwidth allocation. Justice is substantially different from previous approaches to bandwidth allocation. Justice provides a guaranteed minimum allocation to each source, at each link, independent of the behavior of other users. Moreover, Justice is flexible and allows network administrators to efficiently configure the *deserved* relative allocation of each host.

We argue that previously defined notions of per-flow fairness, and the algorithms that have been proposed to enforce them in the Internet, fall short in terms of protecting well-behaved users against their greedy counterparts. In contrast, if Justice is enforced in a network, users are guaranteed a certain level of service, independent of the actions of other users in the network.

We believe that Justice brings a radically new perspective on bandwidth allocation in the Internet. By focusing on per-source bandwidth allocation, and enforceability, Justice lays the foundation for an Internet with more predictable performance, better stability, and more flexibility.

References

1. John Nagle, "On packet switches with infinite storage," RFC 970, Dec 1985.
2. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Symposium proceedings on Communications architectures & protocols*. 1989, pp. 1–12, ACM Press.
3. V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, 4, 1988.
4. Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM TON*, vol. 1, no. 4, 1993.
5. W. Feng, D. Kandlur, D. Saha, and Kang G. Shin, "BLUE: A new class of active queue management algorithms," Tech. Rep. CSE-TR-387-99, 15, 1999.
6. Dong Lin and Robert Morris, "Dynamics of random early detection," in *SIGCOMM '97*, Cannes, France, 1997.
7. Teunis J. Ott, T. V. Lakshman, and Larry H. Wong, "SRED: Stabilized RED," in *Proceedings of INFOCOM*, 1999, vol. 3.
8. Rong Pan, Balaji Prabhakar, and Konstantinos Psounis, "CHOKE, a stateless active queue management scheme for approximating fair bandwidth allocation," in *INFOCOM (2)*, 2000.

9. J. Eriksson, S. Krishnamurthy, and M. Faloutsos, "Justice: An enforceable alternative to fair bandwidth allocation," Tech. Rep., U. California, Riverside, 2004.
10. A. K. J. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," IEEE/ACM TON, vol. 1, no. 3, 1993.
11. H. Zhang, "Wf2q: Worst-case fair weighted fair queueing," IEEE Infocom, 1996.
12. Z. Cao, Z. Wang, and E. Zegura, "Rainbow fair queuing: Fair bandwidth sharing without per-flow state," INFOCOM, 2000.
13. H. Zhu, A. Sang, and S.-Q. Li, "Weighted fair bandwidth sharing using scale technique," Computer Communications, vol. 24, 2001.
14. L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," ACM SIGCOMM 1990, pp. 19-29, 1990.
15. S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," IEEE Infocom 1994, 1994.
16. S. Suri, G. Varghese, and G. Chandranmenon, "Leap forward virtual clock:a new fair queueing scheme with guaranteed delays and throughput fairness," IEEE Infocom 1997, 1997.
17. M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," IEEE/ACM TON, vol. 4, no. 3, 1998.
18. D. Saha, S. Mukherjee, and S. Tripathi, "Carry-over round robin: A simple cell scheduling mechanism for atm networks," IEEE/ACM TON 6 (1998).
19. A. Banchs, "User fair queing: Fair allocation of bandwidth for users," in *Proceedings of INFOCOM*, 2002.
20. P. McKenney, "Stochastic fairness queuing," 1990.
21. I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," SIGCOMM, 1998.
22. Sally Floyd and Van Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM TON*, vol. 3, no. 4, 1995.
23. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Symposium proceedings on Communications architectures & protocols*. 1989, ACM Press.
24. S. Blake et. al., "An architecture for differentiated services," RFC 2475, Dec 1998.
25. P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing.," In RFC 2267, January 1998.

A Novel Approach for Transparent Bandwidth Conservation

David Salyers and Aaron Striegel*

Department of Computer Science and Engineering,
University of Notre Dame, Notre Dame, IN. 46556 USA
dsalyers@nd.edu, striegel@cse.nd.edu

Abstract. In this paper, we present a novel approach, stealth multicast, which offers a practical solution for the adoption of network-level IP multicast. Rather than focusing on a global scale such as with previous approaches, stealth multicast optimizes efficiency on a domain-wise scale. In short, stealth multicast dynamically combines redundant data payloads into virtual groups for multicast transmission across the domain. At the edge of the domain, the packets are converted back to unicast, thus keeping stealth multicast true to its namesake in that neither the user applications nor the external domain are aware of the presence of multicast. We present simulation studies to show the potential of stealth multicast.

1 Introduction

As the Internet has grown and evolved, the demands on the network have shifted from simple connectivity to more sophisticated demands. The point-to-point nature of the Internet has created an increasing presence of redundant data as the applications using the network increase in both scope and scale [1, 2].

A wide variety of techniques have emerged to increase the efficiency of the network. They can be generally classified into two categories: multicast (active) and caching (passive). The first category, IP multicast [3] and application-level multicast (ALM) [4], require that the application and network work together in order to provide maximum efficiency. These technologies are generally hampered by their requirement for global deployment. Although ALM does not have the requirement for network modification, it still requires modifications to the application and can add substantial delay to the transmission.

The second category, is cache based approaches (packet caching [2] and media caching [5, 6]). Although cache-bases approaches do not require the global support of the network or the application, cache based approaches only work for long term redundancy rather than short term (as in multicast).

* This research was supported in part by the National Science Foundation through the grant CNS03-47392.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1219–1230, 2005.

The contrast of multicast and caching-based approaches to bandwidth conservation introduces the motivation for this paper. We propose a new protocol for bandwidth conservation, *stealth multicast*, which provides the deployment simplicity of caching while handling data with short term redundancy normally associated with multicast. Unlike traditional approaches to multicast (IP multicast or ALM) that can require cooperation among various parties using the service (i.e. application support, inter-domain routing), stealth multicast conceals the multicast transport in a domain (or beyond) by dynamically converting packets to and from multicast at the edge of the domain. Hence, stealth multicast frees an ISP from relying on any external participation to yield an immediate, tangible benefit to network performance.

2 Stealth Multicast Overview

Figure 1 illustrates the proposed model and its fundamental concepts wherein a server dispatches information to four separate clients using separate unicasts. The key component of the model is the stealth multicast module which assembles candidate packets into *virtual groups* for multicast transmission across the network. The virtual groups themselves are constructed dynamically[1], based on redundant data payloads from the same source application, and are assembled at the *Virtual Group Detection Manager* (VGDM). Packets that are amenable to multicast (which is determined by the background traffic analysis engine) are queued into multicast groups, and packets that will not benefit from multicast are immediately forwarded without incurring any extra queuing delay. The notion of stealth comes from the fact that the entire process itself is hidden and nearly unnoticeable to the external Internet. The sequence of events is described in more detail below.

First, the application transmits packets with the same data payload to different clients. A digital signature is then created for the data payload (at the VGDM) that uniquely identifies the data payload [2]. The main attributes of the packet (packet size, signature, source IP, and source port) are passed to a background traffic analysis engine. This analysis engine is used to determine if a packet is likely to benefit from multicast. If not, the packet is not queued and is immediately forwarded. If the packet is to be queued, it is queued into a virtual group that shares the same attributes as itself or a new virtual group is created. The packet is then released for transport when a predefined condition is met (group size, timer, etc.). If the redundancy realized is sufficient, the virtual group is transmitted via multicast(PIM-SSM [7]) after selecting the appropriate multicast tree. If not, the packets are forwarded using standard unicast transmission. The unique portions of each packet are preserved as state information. Finally, the packet arrives at the egress point for the domain where the packet is converted back to the original unicast packets using the state information for the

[1] The virtual group itself may have little or no relation to the physical group (actual multicast tree).

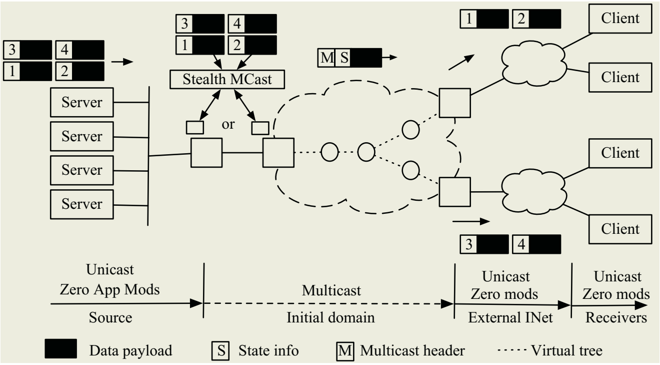


Fig. 1. Stealth Multicast - Overview

virtual group. To the external domain, the unicast packet is indistinguishable from the packet that arrived at the VGDM.

Critical to the ISP, stealth multicast is a sender-driven approach to multicast rather than the receiver-driven approach of network-level IP multicast. The entire multicast process (edge router join/leave) is controlled by the VGDM. Thus, an ISP can easily assess the resource cost and benefit associated with each stealth multicast transmission. Furthermore, stealth multicast is a domain-wise solution rather than an end-to-end solution. Thus, stealth multicast is interested only edge-to-edge transport rather than in end-to-end transport, which improves deployability.

Key Principles: Stealth multicast adheres to two key principles, namely *external transparency* and *QoS impact*. The first principle, external transparency, focuses on deployment. Stealth multicast is application agnostic, and hence, global inter-operability issues and other complexities associated with network-level multicast operation are removed.

The second principle ties into the first principle and into the stealth of the model itself. A significant QoS change may impact the functionality of the applications utilizing the network. Although a positive QoS impact may not necessarily generate criticism, a negative impact on QoS will certainly cause issues with application functionality.

Additional Benefits: First, and most importantly, stealth multicast removes the hurdle of application development to reap a benefit from multicast. Unlike ALM which requires changes to the server and clients, stealth multicast operates transparently to the applications, thus co-existing seamlessly rather than forcing changes to the application. While this does introduce a tradeoff with regards to efficiency, the removal of a dependency on application development increases the immediate appeal of stealth multicast.

Second, the transparency of stealth multicast allows it to be one of the first models to directly address the economic incentives for ISP adoption of multicast.

Unlike existing approaches to multicast where the incentive is almost entirely for the user, the economic benefit of stealth multicast is directable. By varying the location of the VGDM (after the uplink, before the uplink, etc.), an ISP can directly control the benefit of multicast. Furthermore, the sender-driven nature of stealth multicast makes the cost of multicast distribution immediately known at the ingress, thus simplifying shaping as well as offering the opportunity for efficiency-based pricing.

3 Stealth Multicast Operation

For conceptual purposes, the VGDM can be viewed as a collection of COTS hardware dedicated to serving an uplink whose traffic can benefit significantly from stealth multicast. Figure 2(a) shows the components involved once the packet arrives at the VGDM which are discussed below.

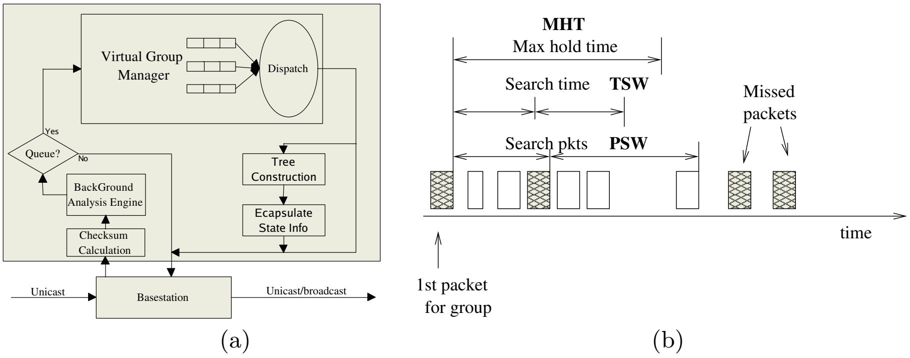


Fig. 2. (a) VGDM - Basic Components (b) Virtual group search settings

Signature Generation: Once a packet arrives at the VGDM, it is uniquely categorized according to its data contents. This is done by generating a signature for the payload of the data packet (this does not include the header information of the packet). The signature is computed using an MD5 checksum on COTS hardware [2][2], which will give a checksum that is sufficient in that no two unique packets of the same size will calculate to the same signature.

Background Traffic Analysis Engine: Once the signature has been calculated, the pertinent packet information (source IP, port, data size, and signature) is sent to the background traffic analysis engine. The background analysis engine is used to track if a specific source IP and port is likely to produce packets that will benefit from stealth multicast. The potential for benefit is determined

[2] Shown to be easily done on a Pentium II 300Mhz machine running Linux 2.0.31 at speeds greater than 45Mb/s when running in user-space.

by keeping track of if the data from a specific source IP and port would have been mulitcasted had it been queued. The engine is then queried to determine if the packet should be queued for stealth multicast or be immediately forwarded because it is not likely to benefit from stealth multicast.

When the analysis engine is queried, there are three possible cases. The first is that the combination of source IP and port have not been seen before. If this is the case, a new entry is created in the analysis engine and the packet is forwarded along under normal unicast transmission. The next possible outcome is that the source IP and port are known not to benefit from stealth multicast; again in this case the packet is immediately forwarded using unicast. The final possible outcome is the source IP and port are known to be amenable to multicast.

3.1 Virtual Group Management

If the packet is known to be a good candidate for stealth multicast, it is passed to the virtual group manager for placement into virtual groups. Three different trigger mechanisms are used to define the performance (additional multicast efficiency) as well as the penalty (queuing delay) introduced by stealth multicast. The goal is to balance these two trade-offs.

The three dispatch triggers (shown in Figure 2(b)) are **MHT**, **TSW**, and **PSW**. *MHT* is the maximum amount of time that a virtual group can be queued, it places are hard upper bound on the impact queuing will have. *TSW* is the inter-packet delay between two packets in a virtual group. Finally, PSW, is a hard limit to the number of packets to scan before a group is released. Once a packet is triggered for release, it is given to the transport mechanism for dispatch via multicast or unicast (insufficient virtual group membership). A more detailed description of the triggers for virtual group release is presented in [8].

Overflow: The memory requirements for the VGDM are quite low. However, in the case of overflow the packets are simply not queued and forwarded as unicast transmissions to their destinations. A more thourough discussion can be found in [8].

3.2 Multicast Transport

If the packet is to be multicast the next dominant issue is how to transport the packet across the domain. While IP multicast operates in a receiver-driven approach, the actual makeup of the receivers in stealth multicast is not known *a priori* and may be highly dynamic depending upon the accuracy of the virtual group detection mechanism. However, this problem is somewhat simplified as multicast transport need only be concerned with transport across the domain whereby the packet is converted back to unicast. Thus, a receiver in the context of a virtual group is the egress point for the domain where conversion occurs rather than the end point (client).

For transport, we propose to employ a dynamic approach whereby multicast groups are created/updated to satisfy all potential egress points for a given source application. In essence, multicast groups are created on a 1:1 ratio with the amenable sources that the VGDM has detected such that each multicast group is a superset of all likely egress points for the source application (see Figure 3). Unlike the broadcast approach whereby all egress points are on the tree, and hence significant bandwidth may be wasted, the egress points are added to the group only as necessary. In the event that an egress point is not yet in the multicast group for the source application, the packet can simply be sent onwards via unicast in the interim.

To avoid any major deployment issues, the dynamic trees are created by coupling extra control messages with PIM-SSM functionality. For all trees originating from stealth multicast, the VGDM is considered the source since it is the initial point for conversion to multicast. We assume that the VGDM knows the identity of all of the egress points (either by configuration or discovery[3]), and that a broadcast tree containing all of the egress points has been created. In order to drive group-wise changes, the VGDM broadcasts control messages containing a list of PIM-SSM commands for the egress nodes to execute. The commands are included via XML to allow for simple inter-operability between a VGDM and an edge router (see [8]). With this capability, the receiver-driven nature of multicast is offered as a sender-driven interface for the VGDM. In addition, the control messages may be augmented to request acknowledgment, QoS-based reservations, and other third party extensions due to the XML basis of the messages.

Egress Point Selection: The notion of dynamically created trees offers several unique design challenges for egress point selection that must be addressed.

- *Virtual Join:* What is the threshold for an egress point to officially become part of the tree? If the VGDM sees only one packet going out a specific client (and its respective egress point) and no future packets, it does not make sense to automatically add the egress point.
- *Virtual Leave:* When does an egress point get purged from the group? Since the VGDM may be inaccurate in detecting virtual groups, individual end points may be missing from the virtual group but yet be found later on. Thus, one must address how to distinguish inaccuracy versus the client actually no longer receiving data from the server application.

Due to the fact that there is no explicit multicast signaling outside of the domain, the makeup of the group itself must be derived from the monitoring of virtual group behavior over time. To that end, we propose to use a windowed history for each end client from the source application. The history window contains a sequence of N binary values whereby a 1 dictates that the client was included in the virtual group and a 0 means the client was absent. Since the virtual group is tracked on the basis of clients (destination IPs) internally rather than egress points, the history is kept on a source-wise basis. A state machine for the egress point is presented in [8].

[3] The notion of a discovery protocol is beyond the scope of the paper.

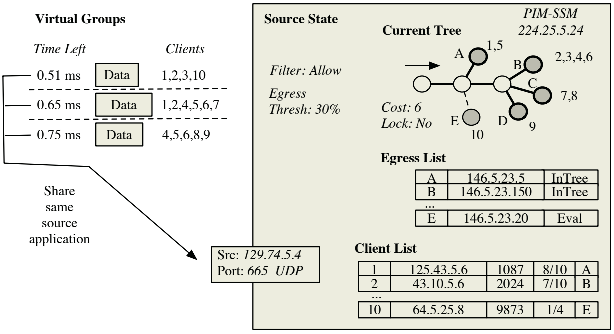


Fig. 3. Virtual groups versus application state

In the event that the multicast tree can be updated instantly, such as with stateless multicast approaches [9], there is no risk for coherency issues. However, given that the multicast control messages need to flow across the domain and follow through with the join or leave procedure, special care must be taken to ensure that changes to the group are restricted. Thus, we employ the concept of a time-based lock whereby once a change is initiated, further changes to the application state are restricted until a sufficient amount of time has passed. In the event an acknowledgment of changes is required, a message from the egress point involved in the change may be solicited by the VGDM.

3.3 State Management

A related component to the transport issue is how to manage the unique portions of state associated with domain. While the transmission of a packet to the appropriate egress points handles the issue of intra-domain distribution, the issues of how to convert the packet back to unicast and where the packet should be converted need to be addressed. For UDP-based applications (the primary beneficiary of stealth multicast), only the destination port and destination IP need to be saved from the packet. For all other fields in the packet, the fields should be the same.

To solve this issue, two approaches may be employed. In the first and simplest approach, the unique pieces of state can be directly included in the packet after the layer 4 (UDP) header. To the core routers in the domain, the additional state information appears as application-level data. When the packet reaches an egress node, the egress node will examine the packet to determine the end destination IDs that it is responsible for and appropriately create the new unicast packets for transmission onwards. Note that it is critical to specify the correct egress node to ensure that multiple egress points do not convert the same end destination IDs.

In the other approach, state information may be distributed amongst the egress points. Using a reliable transport, the VGDM can place the state information at the egress nodes and use only a token to reference the actual state information rather than all of the unique portions of the data. A token must be included to correctly identify end clients as the actual recipients of the information may differ from the stored information. For applications with a semi-static list of clients, the distributed state approach presents an optimal solution as it minimizes the overhead for the transmission of state information. However, the approach does add additional complexity and state storage requirements at the edge router that may not be ideal for all circumstances.

3.4 Other Issues

In addition to the earlier design issues, we address several other issues associated with stealth multicast and its relevance that include the following:

- *Scalability:* Although there are potential concerns in stealth multicast with regards to scalability, the scalability concerns can be mitigated in several respects. First, the ability to correctly detect identical packet payloads at significant line speeds has already been documented via COTS hardware in [2]. Second the amount of storage required at the VGDM is limited due to the fact that only traffic that is likely to be multicast will be queued and the fact that only the first packet in a group needs to be completely stored. In [8] we demonstrate the low requirements for stealth multicast.
- *Practical benefit:* While stealth multicast is well suited for areas with a reasonable amount of redundant traffic, it is not envisioned that VGDMs become ubiquitous at all edge routers. Rather, it is envisioned that VGDMs will be placed at strategic locations in the network rather than via extensive distribution.
- *TCP:* While TCP traffic can work with stealth multicast it is unlike UDP, which is a connectionless send and forget mechanism. TCP involves the possible retransmission of dropped packets and additional state overhead. These features cause TCP traffic, in general, not to be amenable to stealth mulitcasting.

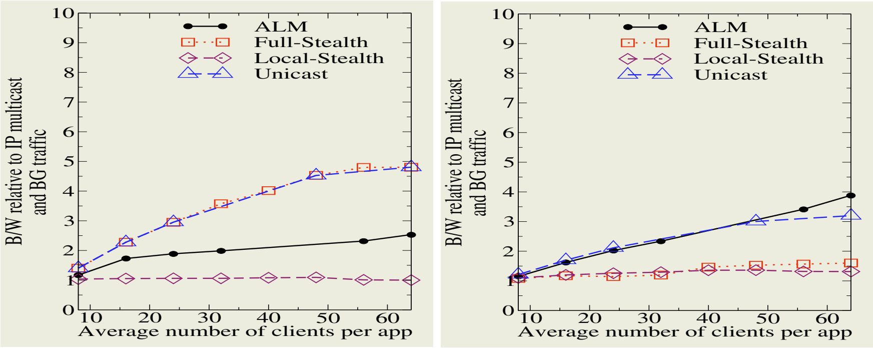
4 Simulation Studies

In this section, we present simulation studies of the stealth multicast architecture. The simulations were conducted using the ns-2 simulator. The purpose of our studies was to examine the performance implications of stealth multicast with regards to other approaches (ALM, unicast, IP multicast) as well as the impact on end user QoS.

The rationale behind our simulations was the following. In the network, Company X hosts an on-line gaming service with applications serving up to 64 clients. The applications are hosted on a set of servers with the clients existing throughout the global Internet. The parameters of the simulation were as follows:

- Random ISP domain (32 core nodes, 16 edge nodes).
- Server Farm (40 source Applications).

- The average number of clients listening to a server application was 32 clients randomly distributed amongst the edge nodes.
- Each join or leave (client joining or leaving) was exponentially distributed with an average inter-arrival event time of 500 ms for all clients in the simulation. The probability of the event being a join or leave was 0.5. The join/leave of the client was not conveyed to VGDM and observations on joins and leaves of the clients were derived using traffic patterns.
- The server applications sent data using UDP packets with an exponentially distributed packet rate of 50 ms and a packet size exponentially distributed with a mean of 500 bytes. For simplicity, the packets were streamed using a constant size unique to each application.
- The settings for the VGDM are listed in [8].



(a) (b)

Fig. 4. Effect of average number of clients with Aggregation on (a) bandwidth - uplink (b) bandwidth - domain

The primary purpose of the simulations was to evaluate the basic principles of the stealth multicast model (impact of queuing, predictability of control parameters, etc.). For the simulations, the bandwidth utilization and end-user QoS were used as the performance metrics. In the simulations, we compared the performance of four distinct models under varying configurations that included:

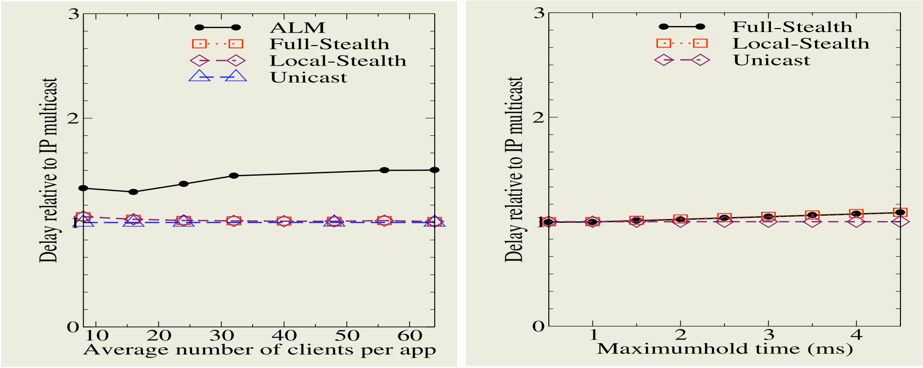
- *Traditional Unicast:* In this model, no stealth multicast is employed.
- *Full Stealth:* In this model, the VGDM is placed at the edge router of the ISP. Traffic must first pass through the customer's uplink before being considered as a candidate for stealth multicast.
- *Local Stealth:* In this model, the VGDM is placed at the edge router of the customer. The traffic can be considered for stealth multicast before being transmitted on the customer uplink.

- *ALM:* A generic version of ALM was used that is loosely based on End System Multicast [10]. Clients for ALM have an asymmetric bandwidth restriction with the ability to support 5 successive downstream connections.
- *IP multicast:* Everything is multicast that can be and no extra overhead B/W or latency is incurred.

All simulation results were normalized to IP multicast in order to compare the relative performance of the competing multicast technologies. In Fig. 4 unicast B/W consumption reaches its peak at 56 clients, this is due to packet loss starting to occur due to the link becoming saturated. Additional simulations can be found in [8].

4.1 Effect of Client Subscriptions, with Aggregation

In this simulation, background traffic (traffic not amenable to stealth multicast) is included. If the performance of stealth multicast is severely affected by background traffic than stealth multicast would be of little practical benefit. The background traffic was generated with simulated UDP and TCP sources. The sources were placed in such a way that all of their packets were detected by the VGDM. Figures 4 and 5(a) shows the performance of the VGDM under these conditions. As can be seen, stealth multicast offers a 3x to 4x improvement over the unicast case. The reason the difference between unicast and stealth multicast is lower is due to the inclusion of the non-multicast traffic. As this traffic is increased the performance ratio will necessarily go down.



(a) (b)

Fig. 5. (a) Effect average number of clients on QoS delay (b) Effect of MHT on QoS delay

Since the VGDM does only queues packets that are amenable to stealth multicast, these results are expected. However, it is possible that background traffic will spread out the arrival of redundant packets sufficiently enough in order to avoid detection by the VGDM. In this case the performance of the VGDM would degenerate into the unicast case, as no packets would be queued and all would be forwarded as standard unicast transmissions. The solution, as noted earlier, is to place the VGDM closer to the sources that are highly amenable to multicast.

4.2 Effect of Detection Parameters

Figure 5(b) shows the effect on queuing delay for the MHT setting. In this simulation PSW and TSW were disabled, allowing for the effect of MHT on the system to be measured. As expected the graph shows a slight linear increase in QoS Delay (it is only a small increase due to unavoidable delay being 33ms and stealth multicast only adds about 1 - 2 ms of delay on average).

However, normally MHT, PSW, and TSW work together in order to minimize the impact on QoS stealth multicast has. This introduces an important point for stealth multicast. In order for the aggregation to have a noticeable impact on queuing delay, the redundant packets need to be sufficiently dispersed so as to continually re-trigger the close proximity rewards of PSW and TSW.

5 Related Work

The most closely related work to stealth multicast lies in [2], whereby caching is applied at the packet level rather than the object/application level. However, unlike stealth multicast, packet caching fares poorly when the redundant traffic exhibits close temporal proximity (such as with streaming media). Furthermore, the work relies on unicast for transport whereas stealth multicast employs detection of virtual groups for multicast transport. Gathercast [11] is another technology related to dynamically increasing the efficiency of the network. However, it deals with a many-to-one (a reverse multicast) relationship where they combine many small packets (such as ACKs) that are for the same destination into one large packet, thus eliminating some overhead.

A short paper presenting the theoretical ideas for stealth multicast was presented in [12]. However this paper moves beyond our initial concept of stealth multicast in several key ways. First, the paper in [12] used DSMCast with tree encapsulation as the method for multicast transport. This caused significant overhead to the data packet to be introduced, thus limiting the benefit of stealth multicast. By using dynamic multicast trees at the VGDM and PIM-SSM signalling as the method for multicast transport, the need for encapsulated state information is removed. This gives a 3x savings in bandwidth which will only increase as the group size increases. Additionally, by using the background analysis engine in order to determine if a packet is likely to be amenable to stealth multicast, the need to queue all packets is removed. This eliminates the penalty normal unicast transmissions would incur in terms of QoS delay while also reducing the memory requirements for the VGDM. While the work in [13] is complementary in that the techniques can be applied to reduce the number of dynamic groups utilized, the work is significantly different in that it assumes an overall IP multicast framework rather than the dynamic conversion of stealth multicast.

6 Conclusions

Stealth multicast offers a novel approach for delivering the beneficial aspects of multicast with the deployment ease associated with cache-oriented approaches. By keeping the presence of multicast hidden from the external Internet, the key problems that have plagued network-level IP multicast are avoided. We believe stealth multicast offers a critical catalyst for spurring the development of large scale group-oriented applications that cannot occur in the current network environment. Stealth multicast offers a controllable and measurable economic benefit for ISPs to incorporate multicast-like efficiency without the complexities traditionally associated with multicast.

With regards to future work, we are developing an open-source prototype of the stealth multicast framework to run experimental studies on live traffic.

References

1. Danzig, P., et al.: A case for caching file objects inside internetworks. In: Proc. of ACM SIGCOMM'93. (1993)
2. Santos, J., Wetherall, D.: Increasing effective link bandwidth by suppressing replicated data. In: Proc. of USENIX. (1998) 213–224
3. Almeroth, K.: The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. IEEE Network **14** (2000) 10–20
4. El-Sayed, A., Roca, V., Mathy, L.: A survey of alternative group communication services. IEEE Network (2003)
5. Mogul, J., et al.: Potential benefits of delta-encoding and data compression for http. In: Proc. of ACM SIGCOMM'97. (1997)
6. Wills, C.E., Mikhailov, M.: Towards a better understanding of Web resources and server responses for improved caching. Computer Networks (Amsterdam, Netherlands: 1999) **31** (1999) 1231–1243
7. Bhattacharyya, S.: An overview of source-specific multicast (ssm). RFC 3569 (2003)
8. Salyers, D., Striegel, A.: A novel approach for transparent bandwidth conservation. Technical Report TR-04-28, Univ. of Notre Dame Comp. Sci. and Engr. (2004)
9. Striegel, A., Manimaran, G.: DSMCast: A scalable approach for diffserv multicast. Computer Networks **44** (2004) 713–735
10. Chu, Y., Rao, S.G., Seshan, S., Zhang, H.: A case for end system multicast. IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast (2002)
11. Badrinath, B., Sudame, P.: Gathercast: The design and implementation of a programmable aggregation mechanism for the internet. In: Proc. of IEEE Int'l Conf. on Computer Communications and Networks (ICCCN). (2000)
12. Striegel, A.: Stealth multicast: A catalyst for multicast deployment. In: Proc. of IFIP Networking, Athens, Greece (2004)
13. Cui, J.H., Maggiorini, D., Kim, J., Boussetta, K., Gerla, M.: A protocol to improve the state scalability of source specific multicast. In: Proc. of IEEE GLOBECOM, Taiwan (2002)

Efficient Bandwidth Sharing in Bus-Based Optical Access Networks

Andre-Luc Beylot[2], Nizar Bouabdallah[1], and Guy Pujolle[1]

[1] LIP6, University of Paris 6, 8 rue du Capitaine Scott, F-75015 Paris, France
nizar.bouabdallah, guy.pujolle@lip6.fr
[2] ENSEEIHT - IRIT/TeSA Lab., 2 rue C. Camichel, BP7122, F-31071 Toulouse
Andre-Luc.Beylot@enseeiht.fr

Abstract. Packet-based optical access ring is becoming a promising solution in metropolitan networks. Its performance depends mainly on how optical resource sharing takes place among the different competing access nodes. In such a network fairness issues are likely to arise between upstream and downstream nodes competing to access a common data channel. This paper provides a new protocol called TCARD (Traffic Control Architecture using Remote Descriptors), based on a preventive reservation scheme, to alleviate the fairness problems. The reservation process consists in forcing the upstream nodes to ensure enough available bandwidth for downstream nodes' transmission purpose. Moreover, we develop analytical models in order to assess the access delay when TCARD is enabled. The analytical and simulation results show how the proposed solution alleviates the performance degradation and the resource sub-utilization while achieving fairness among bus nodes.

1 Introduction

Over the last decade, networks have been witnessing a perpetual growth in data traffic. This shift, driven primarily by the proliferation of internet, has created a rising demand for robust networks, with increasingly high-link capacity and node throughput. Due to the new incumbent challenges, the operators are progressively migrating toward optical core networks taking advantage of the tremendous transmission capacity offered by the optical technology. Thanks to the implementation of wavelength division multiplexing (WDM) in core networks, the relentless need for more capacity may have been satisfied. However, at the boundaries of backbone networks, especially at metropolitan and local area networks (MAN/LAN), an efficient solution for transporting and switching huge amounts of data still needs to be found.

The main goals of next-generation metropolitan networks are to provide high-speed transmissions and the integration of various services (e.g., data, voice, and video) over larger geographic areas. In this regard, three major enabling factors are identified as crucial for the evolution process of metropolitan networks' architecture: optics, packet switching, and protocol convergence.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1231 – 1242, 2005.

To deal with the new requirements of next-generation metro networks, a new architecture named the dual bus optical ring network (DBORN) has been proposed. The DBORN architecture will be described in this article, but for more detailed information the reader is invited to refer to [1]. The performance of such networks depends mainly on how optical resource sharing takes place among the different competing access nodes. Thus, both collision-free transmission and fair sharing of the common bandwidth among ring nodes must be ensured. A medium access control (MAC) protocol is needed to avoid collisions on the individual WDM channels shared among competing nodes. The proposed MAC protocol must consider the case of nonslotted WDM rings. However, even though the MAC protocol enables collision-free transmission over the shared medium, it does not address the inherent fairness issue, which is pronounced in the case of shared medium networks.

In fact, since several source nodes share a common channel, one node may grab all the available bandwidth, and possibly starve the downstream nodes competing to access the same channel. Protocols at various levels (such as MAC or CAC – Call Admission Control) must be introduced to ensure good utilization of the transmission resources and to alleviate fairness problems. In general, fairness control mechanisms limit the transmission of upstream nodes in an attempt to keep enough bandwidth for downstream stations [2]. Although these schemes may be efficient in the case of slotted WDM rings, yet, they do not perform well in the case of asynchronous transmission based architectures like DBORN. We presented in [3] analytical models and simulation results that illustrate the aforementioned issue. Indeed, we noticed that sharing the common bandwidth fairly but arbitrarily among bus nodes, like in slotted WDM ring, doesn't resolve the fairness problem within asynchronous system. In this regard, we demonstrated in [4] the inherent limitations of the token bucket access rate-based algorithm once applied to asynchronous transmission bus-based networks.

As a result, we suggest in this paper a novel control traffic protocol that aims to solve the fairness issue. The proposed solution, the so-called TCARD (Traffic Control Architecture using Remote Descriptors) and presented briefly in [4], is based on a preventive mechanism when granting access to the resource, i.e. free bandwidth is reserved by each node according to the traffic requirements of its downstream partners. In this context, analytical models are developed in an attempt to provide explicit formulas that express the mean access delay of each node of the bus-based optical access network when TCARD is considered.

The remaining parts of this paper are organized as follows. Section 2 emphasizes the MAC context including a description of the ring network along with its main features. Then, section 3 provides a detailed description of the proposed control mechanism TCARD. Analytical models for evaluating the access delay performance of each ring node, when TCARD is considered, are developed in section 4. Then, section 5 validates the accuracy of the proposed models by comparing the analytical results with that obtained by means of simulations. Furthermore we demonstrate how TCARD can achieve fairness and improves the network performances. Finally, conclusions are drawn in section 6.

2 Network Architecture and MAC Design

This section describes the DBORN architecture and the proposed MAC protocol.

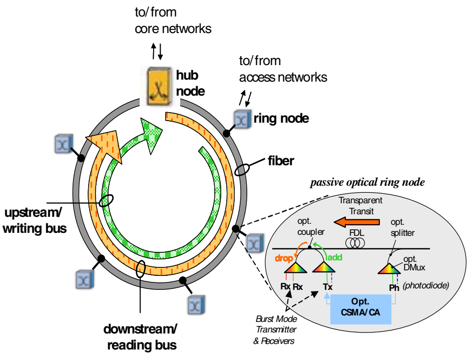


Fig. 1. Overview of DBORN network and node architecture

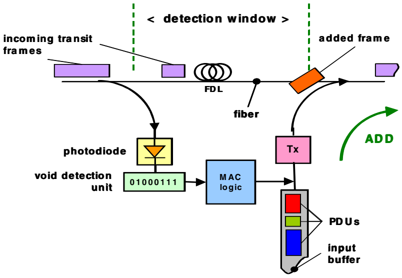


Fig. 2. Schema of the CSMA/CA based MAC of DBORN

2.1 DBORN Architecture

DBORN can be described as a unidirectional fiber split into downstream and upstream channels spectrally disjoint (i.e. on different wavelengths). The downstream bus, initiated at the hub node, is a medium shared for reading purposes, while the upstream bus, initiated in the ring nodes, is a multiple access-writing medium. Note that all the transmissions in the ring are performed between the hub and the ring nodes. Therefore, in the downstream direction, DBORN is a point-to-multipoint network, and in the upstream direction it is a multipoint-to-point network.

Let us consider N nodes placed in the unidirectional ring, as shown in Fig. 1. Each node serves one or more access networks. With regard to the direction from the access networks to the feeder ring, the ring node plays the role of a concentrator. Buffered packets are transmitted along the upstream bus toward the hub. In fact, packets travel along the ring without any electro-optic conversion at intermediate nodes. Thus we deal with the optical transparency property of the transit traffic through intermediate ring nodes. The hub terminates upstream wavelengths and electronically processes the packets. According to its destination, a packet is forwarded either into the backbone or through the downstream bus to reach the ring nodes which are being destined. In the latter case, the ring node picks up a copy of the downstream signal, originating from the hub, by virtue of a splitter in order to recover its corresponding packets. Once split, the main signal is no more processed at the node level and it continues its path to serve the other ring nodes, since a wavelength channel could be shared in reception by several nodes. However, the ring node terminating the copied channel, electronically processes the data packets contained therein, and delivers them to users.

2.2 MAC Design

In terms of logical performance, the main issue is related to the collision-free packet insertion on a shared writing bus. Since the path in transit remains transparent and passive (neither an active optical device nor electronic conversion are employed to handle transit frames), no packet is dropped once transmitted on the ring. Hence, traffic control mechanisms are required at the electronic edge of the ring nodes to avoid collision with transit traffic during its own data emission.

In a fixed-slotted ring system with fixed packet size, void (i.e., slot) filling can be carried out immediately upon detection, since the void duration is a multiple of the fixed packet size duration. The detected void is therefore guaranteed to provide a minimum duration of one fixed packet length. However, in nonslotted ring systems with variable packet length and arbitrary void duration, it is very likely that a collision will occur if a packet is immediately transmitted upon detection of a void's edge.

To meet these requirements, each ring node must retain the upstream traffic flow within the optical layer while monitoring the medium activity. So, as shown in Fig. 2, each ring node first uses an optical splitter to separate the incoming signal into two identical parts: the main transit signal and its copy used for control purposes. With regard to the control part, as in [5], low bit rate photodiodes (ph) –typically 155 MHz- are used to monitor the activity of the upstream wavelengths. Once a free state of the medium is detected, the MAC unit measures the size of the progressing void.

To do so, a Fiber Delay Line (FDL) is introduced on the transit path to delay the upstream flow by one maximum frame duration augmented by the MAC processing time. The length of the FDL is slightly larger than the Maximum Transmission Unit (MTU) size allowed within the network, in order to provide the MAC unit with sufficient time to listen and to measure the medium occupancy. The ring node will begin injecting a packet to fill the void only if the null period is large enough (i.e. at least equal to the size of the packet to be inserted). Undelivered data will remain buffered in the electronic memory of the ring node until a sufficient void space is detected. This way, collision free packet insertion on the upstream bus from the add port is ensured.

However, considering only this basic mechanism, Head Of the Line (HOL) blocking and fairness issues arise. A direct result would be performance degradation for ring nodes that are close to the hub node on the upstream bus. Additional flow control mechanisms must be thus considered, both at the MAC layer and at the upper layers of edge nodes.

3 The Proposed Protocol: TCARD

To regulate the transmission of a node, we can use the TB rate-based control scheme put into action in the case of slotted WDM ring. However, as demonstrated in [4], this same mechanism presents several limitations when dealing with the fairness problem in the case of asynchronous transmission.

We showed in [4] that even if the traffic sourced by upstream nodes, which is regulated by the TB shaper, does not violate the negotiated throughput, it causes unacceptable packet delay to downstream nodes sharing the same channel. Certainly, thanks to TB algorithm the free bandwidth (stated in bit/s) allocated to each node is theoretically sufficient to handle its traffic. However the main issue pertains to the inappropriate distribution of the free bandwidth. This is mainly due to the lack of organization during the emission process in the network. In fact, the mismatch, between the idle period distribution resulting from the upstream nodes' utilization of the medium and the packets' size distribution of the downstream node, often leads to bandwidth waste as well as fairness problems with regard to resource access.

To cope with the aforementioned factors, we suggest a new preventive control mechanism that forces a node to conform to its allocated bandwidth, and it also prevents the random division of the resource capacity. Let us consider N nodes sharing a common unidirectional channel traveling to the hub and let us consider more specifically the i^{th} node of the bus. Unlike the TB algorithm which is based on the i^{th} node traffic descriptor, the TCARD algorithm relies on the specification of the aggregate traffic sourced at the downstream nodes $(i+1,..,N)$. The TCARD mechanism is based on the distribution and the gathering of remote information (descriptors) relative to the traffic requirements contracted by downstream ring nodes sharing the same resource. For each ring node $i(i=1,..,N)$, the aggregate descriptor reflects the traffic needs of its downstream neighbors. This information is then used to constrain the access of the node i to the medium.

The basic operation of such a scheme is simple. To describe it, let us consider an anti-token pool at the node i where anti-tokens are generated at fixed time intervals that correspond to the specified average rate of the aggregate downstream traffic. Unlike TB scheme where each token represents the permission to transmit one bit, each TCARD anti-token prevents the node i from transmission on a detected idle period for a fixed amount of time S. This idle period of size S is reserved for downstream nodes. Following this reservation, an anti-token is removed from the anti-token pool. Packets arriving to the node enter first to the input buffer. Thanks to the MAC protocol, the node listens and measures the void duration present in the medium. If the detected null period is smaller than S, i.e. the anti-token cannot be placed, the void could be used by the node for its own usage purpose as long as the cumulative size of frames to be sent is smaller than the current void size. If the detected null period is equal or greater than S, it is used mainly for the release of one anti-token if the anti-token pool is not empty, otherwise the void can be used by the node for transmission. The arrival of an anti-token to the pool during the transmission of a packet does not preempt the emission process. The anti-token has to wait the packet service completion in order to be served.

Hence, the main idea behind TCARD lies in the preservation of bandwidth (represented by voids) by upstream nodes in order to satisfy downstream nodes' requirements. However, organized and sensible reservation schemes must be applied to fully benefit from the protocol efficiency. A basic rule consists in avoiding random division of the resource leading to inadequacy between the idle period length and the packet size distributions. Indeed, in order to guarantee for a downstream node the ability to transmit packets of maximum size, the length S of a reserved void (which consumes one anti-token) must be at least equal to the MTU size. On the other hand, the maximum length of an idle period that can be reserved is limited by the FDL length adopted in the MAC design. As stipulated earlier, the FDL length used is slightly larger than the MTU size allowed on the network. Therefore, to fulfill the aforementioned requirements the duration S that has to be adopted must be equal to the time required to transmit the MTU on the shared medium.

4 Approximate Analytical Models of TCARD

In this section, we propose a first analysis of the TCARD mechanism. Our aim is to assess the mean waiting time $E[W_i]$ at each node $i(i=1,..,N)$ when TCARD is enabled. To achieve this aim, we will primarily consider the first node. Furthermore, the analysis is extended to the following ring nodes.

4.1 Analysis of the First Node

The first node can be modeled by an M+D/G/1 queue with non-preemptive priority. In fact, the workload for the first queue consists of two classes of job: the anti-tokens, which represent the higher priority class, and the data flows (i.e. packets). Anti-tokens arrive to each node according to a deterministic process with an arrival rate τ_i and require a fixed service time *S*. On the other side, packets arrive according to a Poisson process with rate λ and require a general service time. Recall that the objective is to determine the average waiting time of the lower priority class jobs. This analysis is quite complicated so we propose an accurate approximation of the waiting time of the first node.

In this study, the queue server can be in one of two states: free or busy. The server is busy whenever an anti-token or a packet is present in the system, otherwise the server is free. Let us consider a packet #*j* entering the queue. Two cases are to be distinguished:

a. When the packet arrives, it is the only job of the lower priority class. The packet transmission begins as soon as there are no higher priority jobs (anti-tokens) remaining in the queue. Hence the service time of the packet can be emulated as follows:

   1. If the queue server has gone in a free state since the latest packet transmission (i.e. packet #*j-1*), the emulated service time includes the residual service time of the eventual anti-token already present in the server when the packet #*j* arrives. So packet #*j* may take a waiting time (set up time) before its transmission can start. Note that at most one anti-token may be present in this case, since the anti-token generation rate is lower than the medium throughput. The emulated service time gets over as soon as there are no anti-tokens remaining in the queue since the beginning of packet #*j* transmission.
   2. The queue server is always in a busy state since the packet #*j-1* has been successfully transmitted. In other words, the emulated service of packet #*j-1* is not yet completed. Once this service is completed (i.e. there is no more anti-tokens in the queue) the packet #*j* is taken immediately into service. In this case there is no waiting time or set up time. Indeed, the emulated

service of packet *#j* begins once the packet *#j* transmission starts and gets over as soon as there are no anti-tokens in the queue.

b. In the second case, the packet arrives while other packets are still present in the system. The packet *#j* has to wait its turn to be served. When the service of packet *#j-1* is completed, the packet *#j* is taken into service immediately. In this case, the emulated service time retains the same expression as in the case a-2 and there is not a set up time. Henceforth both cases will be aggregated in the remainder of this study.

In summary, this model behaves as an M/G/1 queue with a set up time [6]. For this model, we first derive the emulated service time distribution as described above. Moreover, we obtain the mean waiting time.

4.1.1 Emulated Service Time with Possible Non Null Set Up Time

It corresponds to the case a-1. As explained earlier, the packet *#j* has to wait during the residual service time of the eventual anti-token still present in the system. Then packet *#j* is transmitted. In this case, the emulated service of packet *#j-1* is already completed. Thus the system is "empty" of customers when packet *#j* arrives even if the server may be occupied by the last generated anti-token. Let T_0 denote the arrival time of the packet *#j*. Due to the memoryless property of Poisson arrivals, we consider that the arrival time A_0 of the last generated anti-token is uniformly distributed over $\left[T_0 - \frac{1}{\tau_1}, T_0\right]$ where τ_1 is the generation rate of anti-tokens. Two cases can be distinguished:

- Case 1: $A_0 < T_0 - S$, i.e. the last generated anti-token has already left the system when the packet *#j* arrives. The set up time is thus null. The packet *#j* transmission begins immediately (at $X_0 = T_0$) and will finish at time $X_0 + S_j$, where S_j is the transmission time of packet *#j*. The following anti-token will arrive at time $A_1 = A_0 + \frac{1}{\tau_1}$.

  — If $X_0 + S_j < A_1$, then the node can transmit a new packet at time $X_0 + S_j$. Hence the service time of the packet *#j* is simply S_j and the set-up time is equal to 0.

  — If $X_0 + S_j \geq A_1$, the anti-token will be sent just after the departure of the packet *#j*, during the time period $[X_0 + S_j, X_0 + S_j + S]$. The next anti-token will arrive at time $A_2 = A_0 + \frac{2}{\tau_1}$. The analysis is iterated until $X_0 + S_j + kS < A_{k+1}$ where $A_{k+1} = A_0 + \frac{k+1}{\tau_1}$. In this case, the set-up time is zero and the emulated service time equals to $S_j + kS$.

- Case 2: $A_0 > T_0 - S$, i.e. the last arrived anti-token is still in the system when the packet *#j* arrives.
  The packet *#j* has to wait at least during the time period $[T_0, A_0 + S]$ before its transmission starts. This will constitute the set-up time of the system. As $S < \frac{1}{\tau_1}$, the last arrived anti-token will leave the queue before the arrival of the following one. Indeed, $A_0 + S < A_1 = A_0 + \frac{1}{\tau_1}$. Thus, the packet *#j* will enter the server at time $X_0 = A_0 + S$ and will be transmitted during the time period $[X_0, X_0 + S_j]$. Then, one falls down in the same study as in the first case with $X_0 = A_0 + S$ instead of $X_0 = T_0$. Hence the service time is given by the expression:

$$S_j + kS + X_0 - T_0 \text{ where } \begin{cases} X_0 + S_j + kS < A_{k+1}, \\ X_0 + S_j + (k-1)S > A_k \\ \text{and with } A_k = A_0 + \frac{k}{\tau_1} \end{cases} \tag{1}$$

  The set up time is $X_0 - T_0$ in this case.

4.1.2 Emulated Service Time with Null Set Up Time

We refer here to the cases a-2 and b. As stated before, the service time expression remains the same in both cases. When the packet *#j* arrives, the emulated service of packet *#j-1* is not yet completed. Thus the system is not "empty" of customers when packet *#j* arrives. The set up time is zero. The emulated service time starts at X_0 as soon as there is no anti-token in the system since the arrival of packet *#j*. Hence, the last anti-token arrived at A_0 with $X_0 - \frac{1}{\tau_1} < A_0 < X_0 - S$. We assume that the arrival of the last anti-token is uniformly distributed between $\left[X_0 - \frac{1}{\tau_1}, X_0 - S\right]$. The packet *#j* will be emitted during the time period $[X_0, X_0 + S_j]$. If during this transmission period, the following anti-token arrives ($A_1 = A_0 + \frac{1}{\tau_1} < X_0 + S_j$), it will be served at time $X_0 + S_j$ and will leave the server at time $X_0 + S + S_i$. In a similar way as previously, this operation is iterated in order to derive the "emulated service time" of a packet *#j*.

4.1.3 The Mean Waiting Time for the First Node

The first node is then analyzed as an M/G/1 queue with set-up time [6]. Let C_0 (resp. C_1) denote the emulated service time when the system is "empty" (i.e. case a-1) (resp. not empty, i.e. cases a-2 and b) at arrival epochs. Let π_0 , be the probability that a packet arrives while the system is "empty" of customers.
It can be shown that:

$$\pi_0 = \frac{1-\lambda E[C_1]}{1+\lambda E[C_0]-\lambda E[C_1]} \tag{2}$$

The mean "emulated service time" $E[C]$ (including set-up times) is consequently equal to:

$$E[C] = \pi_0 E[C_0] + (1-\pi_0)E[C_1] \tag{3}$$

The mean response time can be expressed as follows:

$$E[R] = \pi_0 \left(\frac{2E[C_0]+\lambda E\left[C_0^2\right]}{2(1-\lambda E[C_1])} + \frac{\lambda^2 E[C_0]E\left[C_1^2\right]}{2(1-\lambda E[C_1])^2} \right) \tag{4}$$

The mean waiting time is thus equal to:

$$E[W_1] = E[R] - E[C] + \pi_0 \frac{S^2}{2}\tau_1 \tag{5}$$

Where $\frac{S^2}{2}\tau_1$ is the average residual service time of the eventual anti-token present in the server when a packet arrives while the system is "empty".

Different packet length distributions can be considered. In the present work, we consider packets of variable length (50, 500 and 1500 bytes) more or less representative of the peaks in packet size distribution in Ethernet. The total traffic volume comprises 50% of 1500 Bytes, 40% of 500 Bytes and 10% of 50 Bytes packets size. Let p_i be the probability of the different packet sizes and d_i the corresponding emission time. The transmission time of an anti-token *S* is chosen equal to the emission time of the longest packets.

For each type *i* of packets, we get the probability density function of $C_{1,i}$ (index *i* refers to packets of length d_i):

$$f_{C_{1,i}}(x) = \frac{d_i - k\left(\frac{1}{\tau_1}-S\right)}{\frac{1}{\tau_1}-S}\delta(x-(d_i+(k+1)S)) + \frac{\frac{1}{\tau_1}-S-d_i-k\left(\frac{1}{\tau_1}-S\right)}{\frac{1}{\tau_1}-S}\delta(x-(d_i+kS \tag{6}$$

where δ denote the Delta Dirac function and $k = \left\lfloor \frac{d_i}{1/\tau_1 - S} \right\rfloor$.

Which yields to:

$$\begin{cases} E\left[C_{1,i}\right]=\dfrac{d_i-k\left(\dfrac{1}{\tau_1}-S\right)}{\dfrac{1}{\tau_1}-S}\left(d_i+(k+1)S\right)+\dfrac{\dfrac{1}{\tau_1}-S-d_i-k\left(\dfrac{1}{\tau_1}-S\right)}{\dfrac{1}{\tau_1}-S}\left(d_i+kS\right) \\ E\left[C_{1,i}^2\right]=\dfrac{d_i-k\left(\dfrac{1}{\tau_1}-S\right)}{\dfrac{1}{\tau_1}-S}\left(d_i+(k+1)S\right)^2+\dfrac{\dfrac{1}{\tau_1}-S-d_i-k\left(\dfrac{1}{\tau_1}-S\right)}{\dfrac{1}{\tau_1}-S}\left(d_i+kS\right) \end{cases} \tag{7}$$

In a similar way, the probability density function of $C_{0,i}$ is given by:

$$f_{C_{0,i}}(x)=f_{C_{1,i}}(x)\frac{\dfrac{1}{\tau_1}-S}{\dfrac{1}{\tau_1}}+\frac{\left(x+d_i+kS\right)}{\dfrac{1}{\tau_1}}1_{\{-S<x<0\}} \tag{8}$$

The first two moments of those service times are then equal to:

$$\begin{cases} E[C_0]=\sum_i p_i E[C_{0,i}] & E\left[C_0^2\right]=\sum_i p_i E\left[C_{0,i}^2\right] \\ E[C_1]=\sum_i p_i E[C_{1,i}] & E\left[C_1^2\right]=\sum_i p_i E\left[C_{1,i}^2\right] \end{cases} \tag{9}$$

4.2 Analysis of the Following Nodes

The analytical model described so far handles the first node case. However, it can be easily extended to the following ring nodes without major modifications but still approximations have to be accounted for. To achieve this aim, we assume that the available bandwidth received by each node i is equal to the bandwidth reserved by upstream nodes due to the anti-tokens. Thus the server speed D_i of each node i is considered equal to:

$$D_i = S\tau_{i-1} \text{ with } i>1 \tag{10}$$

Afterwards, we can recursively apply exactly the same method presented in section 4.1.

5 Performance Evaluation

In this section, we evaluate the performance of the proposed fairness control protocol. We consider a ring of 8 nodes sharing the same wavelength that runs at 1 Gbit/s. Each node receives traffic from the access networks to be forwarded toward the hub at a mean rate of 0,1 Gbit/s. Thus the traffic sourced by all the ring nodes represents 80% of the wavelength capacity. The packets arrive according to a Poisson process. The arrival rate of the packets at each node is the same in order to highlight the fairness issues. In this section we assess the access delay and the packet loss rate (PLR) at each ring node. As explained before, the anti-token generation at each node within TCARD is configured to reflect the average amount of traffic expected at downstream nodes. Moreover, the bandwidth reserved for downstream nodes' use purpose is representative of voids of 1500 bytes in order to comply with packets of maximum size. For instance, the TCARD anti-tokens are generated at the first node at a rate of $\tau_1 = (0{,}1 \cdot 7 \cdot 10^9)/(1500 \cdot 8)$ anti-tokens/s.

We first compare the performance of each ring node whether TCARD is enabled or not. Then, we evaluate the accuracy of the proposed analytical model by comparing its results with those obtained from simulations carried out using network simulator 2. Figure 3 depicts the average access delay experienced by packets arriving to each node. Results highlight the fairness issue already discussed. We point out that the performance degradation, when TCARD is disabled, is not due to the medium saturation since the channel occupancy is below 70 %. This is mainly due to the lack of organization during the emission process in the network. In fact, the mismatch, between the idle period distribution resulting from the upstream nodes' utilization of the medium and the packets' size distribution of the downstream node, often leads to bandwidth waste as well as fairness problems with regard to resource access. Recall that the input load is 80%. This difference is simply due to the packet loss resulting from buffer overflow of downstream nodes.

On the other hand, TCARD enables fairness and better use of the resource by sharing efficiently the bandwidth between competing nodes. The mean access delay is around 160 µs for all the nodes. We notice that the performance of the downstream nodes when going closer to the hub is not affected by the upstream nodes. In addition, simulations show that TCARD improves the resource utilization, which increases from 70% to 80%. We point out however that the delay recorded at upstream nodes is slightly increased, with respect to the case where no fairness control mechanism is

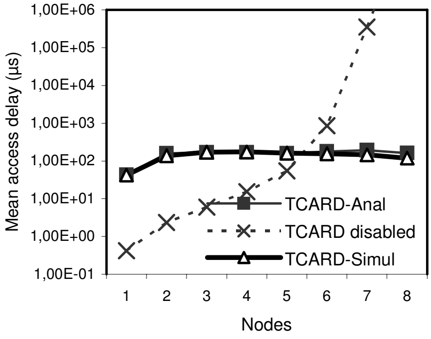


Fig. 3. Mean access delay of the eight-node bus with variable-packet size traffic

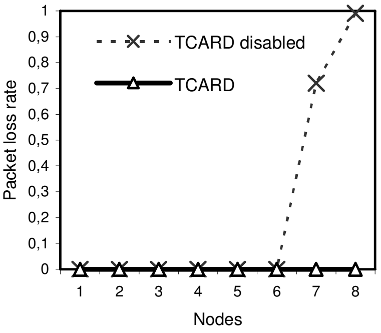


Fig. 4. Packet loss rate of the eight-node bus with variable-packet size traffic

applied, but still remains below 170 μs. This is because TCARD algorithm imposes more constraints on upstream nodes in order to preserve usable bandwidth for downstream ones. The analytical results of access delay when TCARD is enabled reveal a perfect match with the simulation results: analytical results practically coincide with the simulation results. So, the approximate analytical models can achieve high accuracy.

Figure 4 depicts the PLR at each ring node. The capacity of the electronic buffer at each node is set to 1 Mbytes. As expected, when TCARD is disabled, packet loss occurs more and more when approaching the hub due to the node buffer overflow. Indeed, downstream nodes do not find suitable idle period to transmit their packets. In particular, the loss rate registered at node 8 is above 99% in the absence of any control mechanism. In contrast, with TCARD, no packet loss is recorded in the network. In this case TCARD never incurs loss due to its efficient share of bandwidth among nodes.

6 Conclusion

This paper provides the analysis of shared bus network's behavior with asynchronous transmission. We analyzed the system performance in terms of access delay required by each node to inject a packet on the shared medium. The analysis results showed that fairness issues are likely to arise between upstream and downstream nodes. To alleviate the fairness problem, we suggested a new protocol, called TCARD, based on a preventive reservation scheme. We developed analytical models in order to assess the access delay at each node when TCARD is enabled. The proposed models are proven to be highly accurate by comparison with simulation results. The analytical and simulation results showed how the proposed solution alleviates the performance degradation and the resource sub-utilization while achieving fairness among bus nodes.

References

1. N. Le Sauze et al.: A novel, low cost optical packet metropolitan ring architecture. Proc. of ECOC '01, Amsterdam, Netherlands, Vol. 4 (October 2001) 66-67
2. M. A. Marsan, A. Bianco, E. Leonardi, F. Neri, and S. Toniolo: Metaring Fairness Control Schemes in All-Optical WDM Rings. Proc. of INFOCOM '97, Kobe, Japan, vol. 2 (April 1997) 752-760
3. N. Bouabdallah, A. L. Beylot, G. Pujolle: Fairness Issues in Bus-Based Optical Access Networks. Proc. of Networking '04, Athens, (May 2004)
4. N. Bouandallah et al.: Resolving the Fairness Issue in Bus-Based Optical Access Networks. IEEE Commun. Mag., vol. 42, no. 11, (Nov. 2004) pp. 2-8
5. R. Gaudino et al.: RINGO: a WDM Ring Optical Packet Network Demonstrator. Proc. of ECOC '01, Amsterdam, Netherlands, Vol. 4 (September 2001) 620-621
6. H. Takagi: Queueing Analysis Vol I: Vacation and Priority Systems Part I. North Holland (1991)

Cross-Layer Radio Resource Allocation in Packet CDMA Wireless Mobile Networks with LMMSE Receivers

Fei Yu and Vikram Krishnamurthy

Department of Electrical and Computer Engineering,
The University of British Columbia,
2356 Main Mall, Vancouver, BC, Canada V6T 1Z4
{feiy, vikramk}@ece.ubc.ca

Abstract. Most of previous study of radio resource allocation in traditional wireless networks concentrates on network layer connection blocking probability QoS. In this paper, we show that physical layer techniques and QoS have significant impact on network layer QoS. We define a novel concept of cross-layer effective bandwidth and use this to measure the unified radio resource usage taking into account both physical layer linear minimum-mean square error (LMMSE) receivers and varying statistical characteristics of the packet traffic in code devision multiple access (CDMA) networks. We demonstrate the similarity between traditional circuit-switched networks and packet CDMA networks, which enables rich theories developed in traditional wireless networks to be used in packet CDMA networks. Moreover, since both physical layer signal-to-interference ratio (SIR) QoS and network layer connection blocking probability QoS are considered simultaneously, we can explore the tradeoff between physical layer QoS and network layer QoS in packet CDMA networks.

1 Introduction

An efficient resource allocation scheme is crucial for guaranteeing different quality of service (QoS) requirements and fully utilizing the scarce radio resource available in wireless mobile networks. Several schemes have recently been proposed for resource allocation in wireless mobile networks. In [1], the complete sharing (CS) and complete partition (CP) schemes are studied. The CS policy allows all connections equal access to the radio resource at all the time, which will result in maximum usage of the available resource. However, at the same time, it does not provide different network layer QoS (e.g., connection blocking probabilities) to different classes of traffic when traffic load is heavy. The CP policy divides up the available resource into separate sub-pools, and each class of traffic can only access its resource pool. This policy allows for more control of the QoS. In the guard channel scheme [2], [3], a portion of resource is reserved for some important classes (e.g., handoff connections) to provide better QoS to these classes. The fractional guard channel scheme [4] is to admit a less important connection (e.g., new connection) with a certain probability when the system (the number of all ongoing connections) is in certain states. The system state can also be defined as the number

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1243–1254, 2005.

of ongoing new connections. This leads to the new connection bounding scheme [5]. When the resource is not available, some classes of connection requests can be queued instead of being rejected to provide different QoS to different classes [6], [7]. Authors in [8] investigate the comparative performance of different resource allocation schemes.

Although much work has been done in resource allocation of wireless mobile networks, most of previous work concentrates on network layer QoS, blocking probabilities of new and handoff connections, and does not consider physical layer technologies and physical layer QoS. While the decoupling between network layer and physical layer is appropriate for circuit-switched time division multiple access (TDMA) and frequency division multiple access (FDMA) systems, this approach may not be suitable for packet-switched code division multiple access (CDMA) networks. In fact, the interplay between physical layer and network layer plays an important role in CDMA networks with linear minimum-mean square error (LMMSE) multiuser receivers [9]. Unlike the conventional matched filter receivers, the LMMSE receivers take into account the structure of the interference from other users when demodulating a user, and therefore significantly outperform the conventional matched filter receivers [10]. Moreover, unlike traditional circuit-switched networks, future CDMA networks are required to support packet multimedia traffic, which can change the radio resource requirement during a connection's lifetime. Consequently, both cross-layer interplay and packet traffic in CDMA networks complicate the analysis of radio resource allocation schemes as well as hinder the application of rich theories developed in traditional wireless mobile networks [1]-[8] to packet CDMA networks. To the best of our knowledge, analysis of radio resource allocation schemes that considers both CDMA LMMSE physical layer and packet traffic has not been addressed in previous work. For example, the CDMA capacity in [11], [12] is evaluated under the assumption that matched filter receivers are used to demodulate users, and packet traffic is not considered there. In addition, authors in [9], [13] only consider circuit-switched constant bit rate traffic.

In this paper, we study the cross-layer radio resource allocation problem in packet CDMA networks with LMMSE receivers. The novelties of this work are as follows.

1) A novel concept of *cross-layer effective bandwidth* [14] is used to measure the unified radio resource usage taking into account both LMMSE receivers and varying statistical characteristics of the packet traffic in CDMA networks. Based on the concept of cross-layer effective bandwidth.
2) Both physical layer signal-to-interference ratio (SIR) QoS and network layer connection blocking probability QoS can be considered simultaneously in radio resource allocation schemes. Therefore, we can explore the tradeoff between physical layer QoS and network layer QoS in packet CDMA networks.
3) Using numerical examples, we show that physical layer receivers have significant impact on the network layer QoS. We also show that the network layer QoS can be improved significantly if the physical layer SIR QoS can be violated with a small probability. This study reveals a number of interesting observations and provides insights into the radio resource allocation problem from a cross-layer perspective.

The rest of the paper is organized as follows. Section 2 describes the traffic model and CDMA model. Section 3 presents the concept of cross-layer effective bandwidth.

Section 4 discusses cross-layer radio resource allocation schemes. Some numerical examples are given in Section 5. Finally, we conclude this study in Section 6.

2 Model Description

In this section, we formulate the radio resource allocation problem in CDMA networks with LMMSE receivers, as shown in Fig. 1. Packet traffic arrivals request to access the CDMA network. A radio resource allocator decides whether or not to admit a user and allocates radio resource to the user if he/she is admitted. Both network layer blocking probability QoS and physical layer SIR QoS are considered in the resource allocation. The admitted users transmit packet traffic over multi-path fading channels. A LMMSE multiuser detector is used to demodulate each user. The SIR and SIR outage probability evaluated at the LMMSE receivers are passed back to the radio resource allocator. We detail the traffic model and the asymptotic system capacity for CDMA networks with LMMSE receivers in the following.

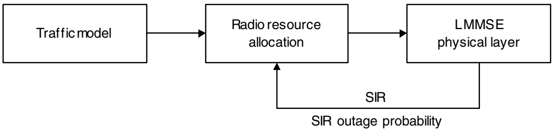


Fig. 1. Cross-layer radio resource allocation in packet CDMA networks with LMMSE receivers

2.1 Traffic Model

Assume there are J classes of traffic in the network. The class $j, j = 1, 2, \ldots, J$, arrival processes of new connections and handoff connections in a cell are Poisson processes with means $\lambda_{j,n}$ and $\lambda_{j,h}$, respectively. We assume that connection holding time for class j connections is exponentially distributed with average value μ_j. Each connection transmits packet traffic in the CDMA network. In order to study the characteristics of the packet traffic and propose the cross-layer effective bandwidth concept in Section 3, we introduce the network layer effective bandwidth concept for a given traffic source in wireline networks, which has been well developed [15] during the last decade. Consider a bufferless communication multiplexer with a single output. There are J classes of input traffic with n_j connections of class j traffic. The aggregate input traffic is $Y = \sum_{j=1}^{J} \sum_{i=1}^{n_j} Y_j^i$, where Y_j^i is the ith class j traffic. Assume that the output capacity is C. The congestion probability of this system is

$$P^{\text{cong}} = P\left\{ \sum_{j=1}^{J} \sum_{i=1}^{n_j} Y_j^i \geq C \right\}. \tag{1}$$

Given the statistical characteristics of traffic sources and their congestion probability requirements, the actual bandwidth that a connection requires lies between its mean rate and its peak rate. This bandwidth is generally referred to as the *effective bandwidth* of the traffic source. Assume that $Y_j[0,t]$ is the amount of work that arrives from a class j source in the time interval $[0,t]$, and $Y_j[0,t]$ has stationary increments. The definition of network layer effective bandwidth of class j traffic is [15]

$$\alpha_j(s,t) = \frac{1}{st} \log \mathbf{E}\left[e^{sY_j[0,t]}\right], s,t \in \mathbb{R}_+, \tag{2}$$

where $\mathbf{E}$ is the expectation, s and t are system parameters defined by the characteristics of the source, its QoS requirements, and the link capacity.

2.2 Fading Channel and Linear Multiuser Detector Physical Layer Model

Signal-to-interference ratio (SIR) is the main QoS measure at physical layer. Evaluating the SIR of LMMSE receivers is difficult due to the interwining of the effects of all signature sequences and received powers of all interferers. Fortunately, recent results [10] show that, the SIR can be closely approximated by an expression that only depends on the transmit powers of all active users as well as the first- and second-order statistics of the channel gain, if we assume that signature sequences of the K users are randomly and independently chosen. In this paper, we use these results in radio resource allocation. The path l of user k is characterized by its estimated average channel gain $\bar{h}_{kl}$ and its estimation error variance ξ_k^2. In a large system (both N and K are large), the SIR for the LMMSE receiver of a user (say, the first one) can be expressed approximately as [10] $\mathrm{SIR}_1 = P_1 \sum_{l=1}^{L} |\bar{h}_{1l}|^2 \eta/(1 + P_1 \xi_1^2 \eta)$, where η is the unique fixed point in $(0,\infty)$ that satisfies $\eta = \left[\sigma^2 + 1/N \sum_{k=2}^{K} \left((L-1)I(\xi_k^2, \eta) + I\left(\sum_{l=1}^{L} |\bar{h}_{kl}|^2 + \xi_k^2, \eta\right)\right)\right]^{-1}$ and $I(\nu,\eta) = \nu/(1+\nu\eta)$. Assume that there are J classes of traffic in the system. An important physical layer performance measure of class j users is SIR_j, which should be kept above the target value ω_j. In [13], it is shown that a minimum received power solution exists such that all users in the system meet their target SIRs if and only if $\omega_j < |\bar{h}_j^i|^2/{\xi_j^i}^2$ and

$$\sum_{j=1}^{J} \sum_{i=1}^{n_j} R_j^i \frac{\Upsilon_j^i}{N} < 1, \tag{3}$$

where $|\bar{h}_j^i|^2$ is the average channel gain of the ith class j user; $|\bar{h}_j^i|^2 = \sum_{l=1}^{L} |\bar{h}_{jl}^i|^2$; n_j is the number of class j users; R_j^i is the number of signature sequences assigned to the ith user of class j to make it transmit at R_j^i times the basic rate (obtained using the highest spreading gain N) and

$$\Upsilon_j^i = (L-1)\omega_j \frac{{\xi_j^i}^2}{|\bar{h}_j^i|^2} + \frac{\omega_j \left(1 + \frac{{\xi_j^i}^2}{|\bar{h}_j^i|^2}\right)}{1+\omega_j}. \tag{4}$$

Note that multi-code CDMA is used in the above model, in which variable bit rate is provided using multiple codes and the SIR requirement of a connection does not change when the bit rate varies [16]. The capacity of the system is restricted by the power control feasibility condition (3). The SIR outage probability in CDMA networks with LMMSE receivers can be expressed as

$$P^{\text{out}} = P\left\{\sum_{j=1}^{J}\sum_{i=1}^{n_j} R_j^i \frac{\Upsilon_j^i}{N} \geq 1\right\}. \tag{5}$$

3 Cross-Layer Effective Bandwidth

3.1 Definition of Cross-Layer Effective Bandwidth

Comparing (5) with (1), we can see the similarity between CDMA networks with LMMSE receivers and wireline networks. From a mathematical point of view, there is a scalar Υ_j^i/N besides the packet traffic R_j^i in (5). It is very interesting to observe that the scalar Υ_j^i/N contains all the information about the physical layer LMMSE receivers. Since the definition of network layer effective bandwidth (2) is useful in deriving the congestion probability (1), we can develop a concept of cross-layer effective bandwidth to derive the SIR outage probability (5). This motivates us to define the cross-layer effective bandwidth of a traffic source as follows:

Definition 1. *Let L denote the number of resolvable paths that each user appears at the receiver,* $|\bar{h}_j^i|^2$ *denote the estimated average channel gain and* ${\xi_j^i}^2$ *denote the channel estimation error variance of the* ith *class* j *connections with a SIR target value* ω_j *in a CDMA system with spreading gain* N*. Let* $R_j^i[0,t]$ *denote the amount of work generated from the* ith *connection of class* j *in the time interval* $[0,t]$*, and* $R_j^i[0,t]$ *is assumed to have stationary increments. The cross-layer effective bandwidth of this connection is*

$$\alpha_j^i(L,N,h,\xi,\omega,s,t) = \frac{1}{st}\log \mathbf{E}\left[e^{sR_j^i[0,t]\Upsilon_j^i/N}\right], L,N \in \mathbb{Z}_+, h,\xi,\omega,s,t \in \mathbb{R}_+, \tag{6}$$

where Υ_j^i *is defined in (4).*

3.2 Properties of Cross-Layer Effective Bandwidth

We derive some properties of the cross-layer effective bandwidth defined in (6). These properties give some insights into the radio resource allocation problem from a cross-layer perspective.

Proposition 1. *If* $R_j^1[0,t],\ldots,R_j^{n_j}[0,t]$ *are* n_j *independent random processes corresponding to the workload from* n_j *class* j *independent connections and* $R_j[0,t]$ *stands from the workload of the multiplexed system,* $R_j[0,t] = \sum_{i=1}^{n_j} R_j^i[0,t]$*, then we have*

$$\alpha_j(L,N,h,\xi,\omega,s,t) = \sum_{i=1}^{n_j} \alpha_j^i(L,N,h,\xi,\omega,s,t). \tag{7}$$

Remark: The cross-layer effective bandwidth for the superposition of independent input processes is the sum of the individual cross-layer effective bandwidths. Therefore, the total cross-layer effective bandwidth of all connections in the system is

$$\alpha(L,N,h,\xi,\omega,s,t)=\frac{1}{st}\log\mathbf{E}\left[e^{\frac{s}{N}\sum_{j=1}^{J}\sum_{i=1}^{n_j}R_j^i[0,t]\Upsilon_j^i}\right]=\sum_{j=1}^{J}\sum_{i=1}^{n_j}\alpha_j^i(L,N,h,\xi,\omega,s,t). \tag{8}$$

This additive property shows the similarity between traditional circuit-switched networks and packet CDMA networks with LMMSE receivers. Therefore, if we allocate each connection with its cross-layer effective bandwidth, rich theories in [1]-[8] can be used to analyze various radio resource allocation schemes in packet CDMA networks.

Proof: See Appendix.

Proposition 2. $\frac{\Upsilon_j^i}{N}\frac{\mathbf{E}\left[R_j^i[0,t]\right]}{t} \le \alpha_j^i(L,N,h,\xi,\omega,s,t) \le \frac{\Upsilon_j^i}{N}\frac{\bar{R}_j^i[0,t]}{t}$.

Remark: The effect of network layer varying statistical characteristics of a connection lies between its mean rate and peak rate in the cross-layer effective bandwidth. Instead of allocating packet multimedia connections with their peak rates or mean rates, we can allocate their cross-layer effective bandwidths in CDMA networks, by which the physical layer QoS can be guaranteed and the network utilization can be increased significantly. We will show this with numerical examples.

Proof: See Appendix.

3.3 Derivation of SIR Outage Probability Using Cross-Layer Effective Bandwidth

In this subsection, we derive SIR outage probability using cross-layer effective bandwidth, which will be used in Section 4. SIR is an important physical layer QoS measure in CDMA networks. However, guaranteeing the SIR of all connections at all time instants will result in low network utilization, especially when the traffic is bursty. Therefore, we use SIR outage probability as a QoS measure in wireless packet CDMA networks. Instead of guaranteeing the SIR at all time instants, we can guarantee the SIR outage probability. This formulation is motivated by the design of packet-switched wireline networks. It is well known [17] that allocating all connections with their peak rates guarantees no packet loss, but results in the lowest utilization and no multiplexing gain. Therefore, most bandwidth allocation schemes in wireline networks allow a small packet loss probability to increase the network utilization [17]. Similarly, since most applications in wireless networks can tolerate small probability of SIR outage, we use SIR outage probability as a QoS measure in wireless packet CDMA networks and keep it below a target value ζ. The SIR outage probability can be estimated by the following well-known *Chernoff bound* [18] approximation

$$P^{\text{out}} = P\left\{\sum_{j=1}^{J}\sum_{i=1}^{n_j}R_j^i\frac{\Upsilon_j^i}{N}\ge 1\right\}\approx e^{\Lambda(1)}, \tag{9}$$

where $\Lambda(v) = \inf_s [s\alpha - sv]$, $\alpha = \sum_{j=1}^{J} \sum_{i=1}^{n_j} \alpha_j^i$, and α_j^i is the scaled logarithmic moment generating function of the instantaneous work load of the ith class j connection in a packet bufferless CDMA system. $\alpha_j^i = \lim_{t \to 0} \alpha_j^i(L, h, \xi, \omega, s/t, t) = (1/s) \log \mathbf{E}\left[e^{sR_j^i \Upsilon_j^i / N}\right]$. The constraint $P^{\text{out}} \leq \zeta$ will be satisfied if the vector $x = (n_1, n_2, \ldots, n_J)$ lies within the admissible set

$$X = \left\{ x \in \mathbb{Z}_+^J : \exp\left\{ \inf_s \left[s \left(\sum_{j=1}^{J} \sum_{i=1}^{n_j} \alpha_j^i - 1 \right) \right] \right\} \leq \zeta \right\}. \qquad (10)$$

The Chernoff bound (9) can be further refined [19] by adding a prefactor. $P^{\text{out}} \approx 1/s^* \sqrt{2\pi \partial^2 (s^* \alpha)/\partial s^2} e^{\Lambda(1)}$, where s^* attains the infimum in (10). The admissible set using the improved bound becomes

$$X = \left\{ x \in \mathbb{Z}_+^J : \frac{1}{s^* \sqrt{2\pi \frac{\partial^2}{\partial s^2} \left(s^* \sum\limits_{j=1}^{J} \sum\limits_{i=1}^{n_j} \alpha_j^i \right)}} \exp\left[s^* \left(\sum_{j=1}^{J} \sum_{i=1}^{n_j} \alpha_j^i - 1 \right) \right] \leq \zeta \right\}. \qquad (11)$$

4 Cross-Layer Radio Resource Allocation in Packet CDMA Networks

Using the concept of cross-layer layer effective bandwidth, we can reduce the complicated packet CDMA networks with LMMSE receivers to traditional circuit-switched networks, and use rich theories developed for traditional networks to analyze various radio resource allocation schemes in packet CDMA networks. In this section, we present the cross-layer global balance equations for general radio resource allocation schemes, from which blocking probabilities and network utilization can be obtained. Then we consider a set of *coordinate convex schemes* that have a product form of the equilibrium probabilities. We emphasize that all of these schemes are not new. However, none of them considers physical layer QoS in previous study. Our contribution is to apply these schemes in CDMA networks with LMMSE receivers using the concept of cross-layer effective bandwidth. Since physical layer QoS, SIR outage probability (9), is considered in cross-layer effective bandwidth, we can study the radio resource allocation problem with both physical layer and network layer QoS.

4.1 Cross-Layer Global Balance Equations

In a packet CDMA network, define the state vector of the system as $x = (n_1, n_2, \ldots, n_J)$, where $n_j, j = 1, 2, \ldots, J$, denotes the number of class j connections in the system. The state space X in the system is defined in (10) or (11). Note that physical layer SIR outage probability QoS constraint, $P^{\text{out}} \leq \zeta$, is used to restrict the state space of the system.

Therefore, different physical layer QoS requirements will result in different state spaces, which have significant impact on the network QoS. As we shall see in Section 5, the network QoS can be improved substantially if a small SIR outage probability is introduced compared to the system in which SIR requirements are guaranteed at all time instants. For each given state $x \in X$, an action $a(x) = (a_1, a_2, \ldots, a_J) \in \{0,1\}^J$ is chosen. If $a_j(x) = 1$, admit a class j connection when the system state is x; if $a_j(x) = 0$, the connection is rejected. The action space is a set of all possible actions, which can be defined as $A = \{a : a \in \{0,1\}^J, j = 1, 2, \ldots, J\}$. The action is done according to a radio resource allocation scheme $u \in \mathcal{U}$, where $\mathcal{U}$ is defined as $\mathcal{U} = \{u : X \to A\}$. $\{x(t), u\}_{t \in \mathbb{R}_+}$ is a Markov process under each radio resource allocation scheme. Let $\pi_u(x)$ denote the equilibrium probability that the system is in state x under scheme u. Define $e_j \in \{0,1\}^J$ as a row vector containing only zeros except for the jth component, which is 1. $x + (-)e_j$ corresponds to an increase (decrease) of the number of class j connections by 1. The global balance equations for the Markov Chain under scheme u are [20]

$$\sum_{j=1}^{J} \pi_u(x-e_j)\lambda_j a_j(x-e_j) + \sum_{j=1}^{J} \pi_u(x+e_j)\mu_j(n_j+1) = \sum_{j=1}^{J} [\lambda_j a_j(x) + \mu_j(n_j+1)]\pi_u(x), \tag{12}$$

where $x \in X, \lambda_j$ and μ_j are class j connection arrival and departure rates, respectively. These global balance equations can be solved using any linear equation procedure, such as Jacobi and Gauss-Seidel methods. Once the equations are solved, network layer blocking probability QoS, can be directly calculated. The blocking probability for a class j connection is

$$P_j^b = \sum_{n \in X_j} \pi(n). \tag{13}$$

where $X_j \subseteq X$ is the set of states that system will move out of X with addition of one connection of class j. This approach is general enough to be applicable to a variety of radio resource allocation schemes.

As the cardinality of X becomes large, the computation complexity of solving the global balance equations is extensive. It is very difficult, if not impossible, to get feasible solutions in real networks due to the problem of large dimensionality. In the following, we consider a set of coordinate convex schemes that have a product form of the equilibrium probabilities.

4.2 Coordinate Convex Schemes

The coordinate convex schemes form several important resource allocation schemes, such as complete sharing, complete partitioning and threshold schemes. It is shown in Chapter 4 of [20] that their equilibrium probabilities have a product form. The name coordinate convex scheme comes from the concept of *coordinate convex set*. A coordinate convex scheme is characterized by a coordinate convex set, which is any nonempty set $\Delta \subseteq X$ with the following property: if $x \in \Delta$ and $n_j > 0$ then $x - e_j \in \Delta$. In a coordinate convex scheme associated with coordinate convex Δ, a connection arrival is admitted to the system if and only if the system state remains in Δ after the admission. The equilibrium probabilities of the system can be obtained from the the theory of multiservice loss networks.

$$\pi(n) = \begin{cases} \pi_0 \prod_{j=1}^{J} \frac{(\lambda_j/\mu_j)^{n_j}}{n_j!}, & \text{if } n \in \Delta, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

where π_0 is a normalization constant,

$$\pi_0 = \frac{1}{\sum_{n \in \Delta} \prod_{j=1}^{J} \frac{(\lambda_j/\mu_j)^{n_j}}{n_j!}}. \tag{15}$$

5 Numerical Results and Discussions

In this section, we illustrate the performance of the proposed approaches by numerical examples. The numerical values used for the system parameters in the numerical examples are given in Table 1. A CDMA system with system bandwidth 5 MHz and spreading gain $N = 512$ is considered. There are two classes of traffic, voice and MPEG video. 20% of the traffic arrivals are video connections. The service rates are μ_1 and μ_2. The voice traffic is modeled as an ON/OFF process. The packet transmission rate in state ON is 15 kbps corresponding to an equivalent spreading gain 256. The rate from ON to OFF is the same as the rate from OFF to ON, which is 0.4, for the voice traffic. A Markov model with two states, NORMAL and BURST, is used for the MPEG traffic [21]. The rate from state NORMAL to state BURST is 0.024 and the rate from state BURST to state NORMAL is 0.076. The data transmission rate in state NORMAL is 30 kbps corresponding to an equivalent spreading gain 128 and the data transmission rate in state BURST is two times of that in state NORMAL.

We compare the admissible region of a packet CDMA network using LMMSE receivers with that of the traditional scenario in which all users are demodulated by matched filters. Two values of the number of resolvable path are considered, $L = 1$ and $L = 5$. SIR outage is not allowed in this example. Fig. 2 compares the admissible regions when using matched filters vs. using LMMSE receivers. We notice a significant gain in the admissible region when LMMSE receivers are used, such illustrating that physical layer receivers have a significant impact on network layer QoS. We further show this by presenting the connection blocking probabilities of voice in Fig. 3. It is observed that the connection blocking probabilities of voice can be decreased substantially in the LMMSE cases, which illustrates the importance of considering physical layer techniques in the radio resource allocation problem in packet CDMA networks. We also observe that the simulation results roughly agree with those from the analysis.

Using the concept of cross-layer effective bandwidth, we can study the effects of physical layer SIR outage probability QoS on network layer connection blocking probability QoS. We compare three radio resource allocation schemes, peak rate, mean rate and cross-layer effective bandwidth with a small SIR outage probability. In the peak rate allocation scheme, all connections are allocated with their peak rates to guarantee

Table 1. Parameters used in numerical examples

Parameter	Notation	Value
target SIR for voice traffic	ω_1	7 dB
estimated average channel gain for voice traffic	$\lvert\bar{h}_1\rvert^2$	1
channel estimation error variance for voice traffic	ξ_1^2	0.02
data transmission rate in state ON for voice traffic	R_1	15 kbps
service rate for voice traffic	μ_1	0.005
target SIR for video traffic	ω_2	10 dB
estimated average channel gain for video traffic	$\lvert\bar{h}_2\rvert^2$	1
channel estimation error variance for video traffic	ξ_2^2	0.05
data transmission rate in state NORMAL for video traffic	R_{21}	30 kbps
data transmission rate in state BURST for video traffic	R_{22}	60 kbps
service rate for video traffic	μ_2	0.004

the SIRs of all connections at all time instants, and the SIR outage probability is zero. [9] and [13] are examples of the peak rate scheme. In the mean rate allocation scheme, each connection is allocated with its mean rate. Figs. 4, 5 show the admissible regions and the video connection blocking probabilities, respectively. We can see that physical layer SIR outage probability QoS has significant effects on network layer blocking probability QoS. Although peak rate allocation approach can guarantee physical layer SIR requirements at all the time, it results in the smallest admissible region and the highest blocking probabilities. An interesting observation is that, with a small SIR outage probability 0.005, the cross-layer effective bandwidth approach can increase the admissible region and decrease the blocking probabilities substantially. The mean rate allocation scheme can further increase the admissible region. However, physical layer SIR outage probability cannot be guaranteed in this scheme, which is more than 50%.

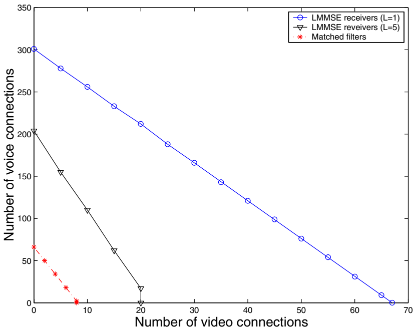


Fig. 2. Admissible regions of different physical layer receivers (SIR outage probability = 0)

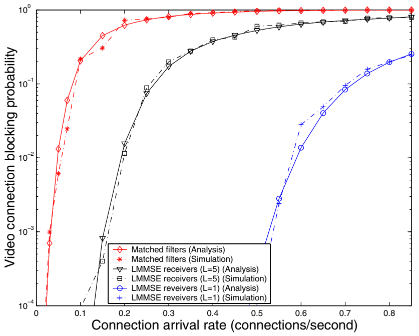


Fig. 3. Voice connection blocking probabilities of different physical layer receivers (SIR outage probability = 0)

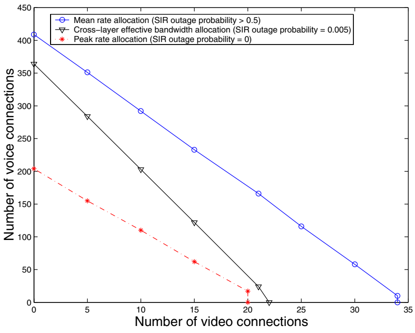


Fig. 4. Admissible regions of different physical layer QoS

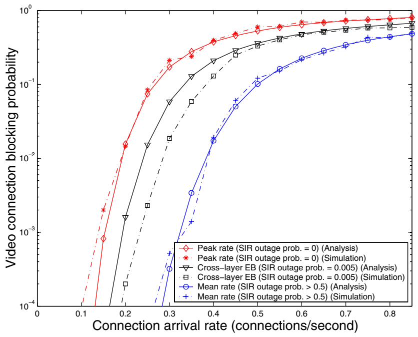


Fig. 5. Video connection blocking probabilities of different physical layer QoS

6 Conclusions

We have studied the radio resource allocation problem in packet CDMA wireless mobile networks from a cross-layer perspective. A novel concept of cross-layer effective bandwidth was used for variable bit rate multimedia traffic in packet CDMA networks. Using this concept, we can have a unified measure of resource usage taking into account both physical layer linear minimum mean square error (LMMSE) multiuser receiver structures and varying statistical characteristics of packet traffic. We have shown that physical layer techniques and QoS have significant effects on network layer blocking probability QoS. Substantial performance gain can be achieved using linear minimum-mean square error (LMMSE) receivers over the scenario in which matched filters are used. It was also observed that network layer QoS can be improved significantly if physical layer QoS can be violated with a small probability.

References

1. B. Epstein and M. Schwartz, “Reservation strategies for multi-media traffic in a wireless environment,” in *Proc. IEEE VTC’95*, vol. 1, (Rosemont, Illinois), pp. 165–169, July 1995.
2. D. Hong and S. Rappaport, “Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedures,” *IEEE Trans. Veh. Technol.*, vol. VT-35, pp. 77–92, Aug. 1986.
3. S. Wu, K. Y. M. Wong, and B. Li, “A dynamic call admission policy with precision QoS guarantee using stochastic control for mobile wireless networks,” *IEEE/ACM Trans. Networking*, vol. 10, pp. 257–271, Apr. 2002.
4. R. Ramjee, D. Towsley, and R. Nagarajan, “On optimal call admission control in cellular network,” *Wireless Networks*, vol. 3, pp. 29–41, Mar. 1997.
5. Y. Fang and Y. Zhang, “Call admission control schemes and performance analysis in wireless mobile networks,” *IEEE Trans. Veh. Technol.*, vol. 51, pp. 371–382, Mar. 2002.
6. Y. Lin, S. Mohan, and A. Noerpel, “Queueing priority channel assignment strategies for handoff and initial access for a pcs network,” *IEEE Trans. Veh. Technol.*, vol. 43, pp. 704–712, Aug. 1994.

7. V. Lau and S. Maric, “Mobility of queued call requests of a new call-queueing technique for cellular systems,” *IEEE Trans. Veh. Technol.*, vol. 47, pp. 480–488, May 1998.
8. B. Li, L. Li, B. Li, and X. Cao, “On handoff performance for an integrated voice/data cellular system,” *ACM/Baltzer Wireless Networks*, vol. 9, pp. 393–402, July 2003.
9. S. Singh, V. Krishnamurthy, and H. V. Poor, “Integrated voice/data call admission control for wireless DS-CDMA systems with fading,” *IEEE Trans. Signal Proc.*, vol. 50, pp. 1483–1495, June 2002.
10. J. Evans and D. N. C. Tse, “Large system performance of linear multiuser receivers in multipath fading channels,” *IEEE Trans. Inform. Theory*, vol. 46, pp. 2059–2078, Sept. 2000.
11. J. Evans and D. Everit, “On the teletraffic capacity of CDMA cellular networks,” *IEEE Trans. Veh. Technol.*, vol. 48, pp. 153–165, Jan. 1999.
12. C. Lindemann, M. Lohmann, and A. Thummler, “Adaptive call admission control for QoS/revenue optimization in CDMA cellular networks,” *ACM/Baltzer Wireless Networks*, vol. 10, pp. 457–472, July 2004.
13. C. Comaniciu and H. V. Poor, “Jointly optimal power and admission control for delay sensitive traffic in CDMA networks with LMMSE receivers,” *IEEE Trans. Signal Proc.*, vol. 51, pp. 2031–2042, Aug. 2003.
14. F. Yu and V. Krishnamurthy, “Effective bandwidth of multimedia traffic in packet wireless CDMA networks with LMMSE receivers – a cross-layer perspective,” *IEEE Trans. Wireless Commun.*, to appear, 2005.
15. F. Kelly, “Notes on effective bandwidths,” in *Stochastic Networks: Theory and Applications* (F. Kelly, S. Zachary, and I. Ziedins, eds.), pp. 141–168, Oxford Press, 1996.
16. C.-L. I and R. D. Gitlin, “Multi-code CDMA wireless personal communications networks,” in *Proc. IEEE ICC’95*, (Seattle, WA), June 1995.
17. M. Schwartz, *Broadband Integrated Networks*. Prentice Hall, 1996.
18. H. Chernoff, “A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observation,” *Ann. Math. Statist.*, vol. 23, pp. 493–507, 1952.
19. R. Bahadur and R. Rao, “On deviations of the sample mean,” *Ann. Math. Statist.*, vol. 31, pp. 1051–1027, 1960.
20. K. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer-Verlag, 1995.
21. J. Pechiar, G. Perera, and M. Simon, “Effective bandwidth estimation and testing for markov sources,” *Performance Evaluation*, vol. 48, pp. 157–175, 2002.

Appendix

Proof (Proposition 3.1). :

$$\alpha_j = \frac{1}{st}\log \mathbf{E}\left[e^{s\sum_{i=1}^{n_j} R_j^i[0,t]\Upsilon_j^i/N}\right] = \frac{1}{st}\log \mathbf{E}\left[\prod_{i=1}^{n_j} e^{sR_j^i[0,t]\Upsilon_j^i/N}\right]$$

$$= \frac{1}{st}\log\left(\prod_{i=1}^{n_j}\mathbf{E}\left[e^{sR_j^i[0,t]\Upsilon_j^i/N}\right]\right) = \sum_{i=1}^{n_j}\frac{1}{st}\log \mathbf{E}\left[e^{sR_j^i[0,t]\Upsilon_j^i/N}\right] = \sum_{i=1}^{n_j}\alpha_j^i.$$

Proof (Proposition 3.2). : Using Jensen’s inequality,

$$\alpha_j^i = \frac{1}{st}\log \mathbf{E}\left[e^{sR_j^i[0,t]\Upsilon_j^i/N}\right] \geq \frac{1}{st}\mathbf{E}\log\left[e^{sR_j^i[0,t]\Upsilon_j^i/N}\right] = \frac{\Upsilon_j^i}{N}\frac{\mathbf{E}\left[R_j^i[0,t]\right]}{t}.$$

$$\alpha_j^i = \frac{1}{st}\log \mathbf{E}\left[e^{sR_j^i[0,t]\Upsilon_j^i/N}\right] \leq \frac{1}{st}\log\left[e^{s\bar{R}_j^i[0,t]\Upsilon_j^i/N}\right] = \frac{\Upsilon_j^i}{N}\frac{\bar{R}_j^i[0,t]}{t}.$$

Stabilization of Contention-Based CDMA Ranging Channel in Wireless Metropolitan Area Networks

Jeong-Jae Won[1], Choong-Ho Cho[2], Hyong-Woo Lee[3], and Victor C.M. Leung[1]

[1] Department of Electrical and Computer Engineering,
The University of British Columbia,
2356 Main Mall, Vancouver, BC, Canada V6T 1Z4
{wonjj,vleung}@ece.ubc.ca
[2] Department of Computer and Information Science, Korea University
[3] Department of Electronics & Information Engineering, Korea University
{chcho,hwlee}@korea.ac.kr

Abstract. The current IEEE 802.16a and 802.16d standards of wireless metropolitan area networks specify a ranging channel in the OFDMA physical layer that employs a set of CDMA codes for ranging and bandwidth requests. The ranging channel is contention-based and inherently unstable. In this paper, we propose two stabilization algorithms to enable efficient utilization of the ranging channel at close to the theoretical throughput limit, and analyze their performance using a continuous time Markov chain M/M/1 model. We show how to estimate system parameters such as the number of backlogged users, arrival rate and the first exit time for the critical. Based on these parameters, we present two methods for channel stabilization, for the cases that the number of ranging codes per frame is fixed and adjusted dynamically, respectively. We then present simulation results to show that, by restricting the actual arrival rate or dynamically adjusting the number of ranging codes, the ranging channel can be stabilized under all traffic conditions.

Keywords: Wireless metropolitan area network (WMAN), OFDMA, CDMA, Ranging channel, Ranging code, System stability.

1 Introduction

The new IEEE 802.16a standard [1] and 802.16d draft standard [2] for wireless metropolitan area networks (802.16 WMANs) specify a protocol that among other things support real-time applications such as voice and video, provides broadband connectivity without requiring a direct line of sight between the subscriber station (SS) and the base station (BS), and support hundreds if not thousands of SSs from a single BS. A SS can be a fixed terminal or a mobile terminal (MT). The standard will help accelerating the introduction of broadband wireless equipment into the market, speeding up last-mile broadband deploy-

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1255–1266, 2005.

ment worldwide by enabling service providers to increase system performance and reliability while reducing their equipment costs and investment risks [3].

In this paper, we focus on the contention-based ranging/bandwidth request channel in 802.16 WMANs, which employs a set of CDMA ranging codes over the OFDMA physical layer. We refer to this system as OFDMA-CDMA multichannel ALOHA (OC-ALOHA) since this system has properties similar to slotted ALOHA. Subsets of ranging codes are allocated in the uplink channel descriptor (UCD) for initial ranging, periodic ranging and bandwidth requests. In [1], [2], an SS transmits a ranging code chosen at random from the assigned subset of codes in one of the slots, again chosen at random from the assigned set of slots. If no other SS transmits the same code in the same slot, the code will be successfully received by the BS. Otherwise a collision occurs and all the transmissions involved are destroyed. The SSs involving in the collision back off and retransmit.

As the access procedure is similar to slotted ALOHA [4], [5] which is inherently unstable, a relatively large set of ranging codes and ranging slots (referred here as code slots) is usually allocated for ranging and bandwidth requests. This allocation for ranging channel may also be altered by various reasons such as the dynamic adjustments of the sizes of uplink sub-frames inserted in uplink map (UL-MAP). Although the performance of existing random access protocols such as GPRS in [4], [5], WLAN in [5], [7] and HIPERLAN in [8], [9] has been widely studied, analysis of the contention-based CDMA ranging channel in 802.16 WMANs is still an open problem. In our previous work [6], the performance of the ranging channel has been evaluated by a continuous time Markov chain (CTMC) M/M/1 model which is approximated by the discrete time Markov chain (DTMC) model, as the exact DTMC model can become intractable. This analytical model allows the unique characteristics of the ranging system to be studied in detail.

Based on the CTMC M/M/1 model presented in [6], we propose in this paper two novel stabilization algorithms that enable the efficient utilization of the contention-based CDMA ranging channel in 802.16 WMANs. We first present a performance evaluation using the CTMC M/M/1 model to show how system parameters such as the number of backlogged users, arrival rate and the first exit time (FET) for the critical state can be estimated and utilized for system stabilization. Then, we present two methods for system stabilization, for the cases that the number of ranging codes per frame is fixed and adjusted dynamically, respectively. In the former case, the rate of ranging and bandwidth code requests that can be transmitted in the next frame is established based on the estimated system parameters. In the latter case, the number of ranging codes per frame is dynamically adjusted based on the estimated system parameters. We further present simulation results to how that the proposed algorithms can always maintain the stability of the ranging channel even if its normalized throughput approaches the theoretical maximum value of 36% for the slotted ALOHA channel.

This paper is organized as follows. In section 2, we review the structure of the ranging channel and the ranging procedure specified in [1], [2]. In section 3, we present the channel stabilization algorithms and analyze their performance in terms of estimated arrival rate, estimated critical state and the estimated number of backlogged users. In section 4, we present the simulation results in comparison with the Pseudo-Bayesian algorithm presented in [4], [10]. In section 5, we draw some conclusions.

2 Contention-Based CDMA Ranging Subchannel

The medium access control (MAC) specifications in [1], [2] define a single ranging channel over the uplink (which we refer as the OC-ALOHA ranging channel) composed of one or more pairs of adjacent subchannels, where the index of the numbered subchannel of each pair is given as shown in Fig. 1. The indices of the subchannel pairs that compose the ranging channel are specified in the uplink message. Transmissions may collide on this ranging channel. To effect a ranging transmission, a SS randomly chooses one CDMA ranging code from among the appropriate sets specified for initial ranging, periodic ranging and bandwidth request.

The initial ranging transmission is sent by any SS that wants to synchronize to the system channel for the first time. An initial ranging transmission is performed during consecutive OFDMA symbols. The same CDMA ranging code is transmitted on the ranging channel during each symbol, with no phase discontinuity between two adjacent symbols. Periodic ranging transmissions are sent periodically to adjust system parameters such as timing and power level. SSs also send bandwidth request transmissions to request uplink channel allocations from the BS. These latter transmissions shall be sent only by the SSs that have already synchronized to the system. To perform either a periodic ranging or bandwidth request transmission, the SS shall modulate one CDMA ranging code on the ranging channel for a period of one OFDMA symbol. Ranging channel is dynamically allocated by the BS MAC and indicated in the UL-MAP message.

The procedures of ranging request are described in Fig. 2. The SS, after acquiring downlink synchronization and uplink transmission parameters, randomly

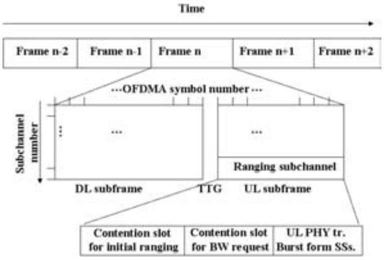

Time
Frame n-2
Frame n-1
Frame n
Frame n+1
Frame n+2
···OFDMA symbol number···
Subchannel number
...
...
...
Ranging subchannel
DL subframe
TTG
UL subframe
Contention slot for initial ranging
Contention slot for BW request
UL PHY tr. Burst form SSs.

Fig. 1. The structure of OFDMA MAC frame

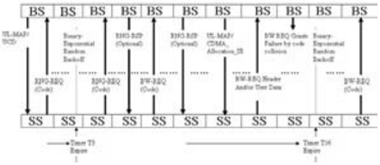


Fig. 2. The procedure of ranging code requests

chooses (a) a ranging slot (with the use of a truncated binary exponential backoff algorithm to avoid possible re-collisions) as the time to perform the ranging, and (b) a ranging code (from the subset of CDMA codes allocated for initial ranging), and sends it to the BS in the selected ranging slot. If no other SS has transmitted the same code in the same slot, the code will be successfully received by the BS. As the BS cannot identify the SS that has sent the CDMA ranging request, upon successfully receiving a CDMA ranging code, the BS broadcasts a ranging response (CDMA_Allocation_IE) message that advertises the received ranging code as well as the ranging slot (i.e., OFDM symbol number, subchannel, etc.) where the CDMA ranging code has been received. This information enables the SS that has successfully sent the ranging code to identify itself. The CDMA_Allocation_IE message contains all the needed transmission adjustments (e.g., time, power and possibly frequency corrections) that the SS should perform, and a status notification. The CDMA_Allocation_IE message also allocates uplink channel resource for the successful SS to transmit its bandwidth request information to the BS. In the unlikely event that the BS successfully captures a code even if two or more SS transmits the same code in the same slot, those SSs will transmit their bandwidth requests in the same allocated uplink channel resulting in a collision. In this case, all the SSs involved in the collision can repeat the bandwidth request procedure using a backoff algorithm [1], [2]. Due to the rare occurrence of this event, we do not consider the possibility of code capturing in this paper.

Upon receiving a CDMA_Allocation_IE message with a "continue" status, the SS waits for expiration of its periodic ranging timer before continuing the ranging process in the same manner as the initial ranging, but with a ranging code randomly chosen from the subset of codes allocated for periodic ranging. In the OFDMA ranging mechanism, the periodic ranging timer is controlled by the SS rather than the BS. After a contention transmission for ranging or bandwidth request, the SS waits for a Data Grant Burst Type IE in a subsequent map (or waits for a RNG-RSP message for initial ranging). Once received, the contention resolution is complete. The SS considers the bandwidth request lost if no data grant has been given within timeout T16 (or no response within timeout T3 for initial ranging) as shown in Fig. 2. The SS shall then increase its backoff window by a factor of two, as long as it is less than the maximum backoff window. The SS shall randomly select a number within its new backoff window and repeat the deferring process described above.

3 Performance Models

To enable theoretical analysis of the performance of request transmissions over the OC-ALOHA ranging channel, we present a CTMC M/M/1 model initially developed in [6] that approximates the DTMC model [11]. This section also considers how measures such as number of backlogged users, arrival rate and the critical state estimation can be utilized for system stability. To provide this stability, we apply our model to derive two algorithms respectively for cases in which the number of code slots in the ranging channel per frame is fixed or can be dynamically readjusted.

3.1 Assumptions

As shown in Fig. 1, a ranging channel consists of two sub-channels and several symbol slots or ranging slots in the OFDMA uplink MAC frame, each divided into a number of ranging codes or ranging code slots. Each code slot accommodates transmission of a CDMA ranging code selected from the subset of codes specified for the intended ranging operation.

As aforementioned, we consider CDMA ranging code requests such as initial ranging, periodic ranging and bandwidth requests. Assume first that the total number of requests generated by all SSs behaves according to a Poisson process with rate λ, such that the probability of k requests being generated per frame amounts to

$$p[k] = \frac{e^{-\lambda}\lambda^k}{k!},\ k = 0, 1, 2, ... \tag{1}$$

where λ is the average number of ranging requests per frame. This is a reasonable assumption as the SSs generally act independently of each other.

If two or more ranging request codes are sent by different SSs using the same ranging code in the same ranging slot in a OFDMA MAC frame, then there will be a collision and the BS will obtain no information about the sources of the requests. If one SS sends a ranging code request in a given ranging slot, the ranging request is correctly received.

To simplify the evaluations of the behavior of the ranging sub-channel, we assume immediate feedback over the downlink channel. This allows the use of a single Markov chain rather than a number of independent Markov chains in the analysis. As this work focuses on the performance of the MAC layer, we assume that the physical transmission is perfectly received if it does not encounter a collision. More specifically, this assumes that the multiple access interference (MAI) between CDMA codes, does not result in transmission errors.

3.2 Mean Delay Time and First Exit Time Analysis

In order to find the access delay of requests in the ranging channel, we define X_i as the number of backlogged users at the end of the i^{th} frame. Then sequence of X_i's forms a Markov chain. As noted in [12], the analysis encounters numerical problem as the number of time slots (corresponding to the code sots in this paper) per frame for packet transmissions increases. Here the number of code slots (n) is defined as the product of the number of codes and the number of slots available in one frame. As the number of code slots (n) increases, the number of successful requests per frame asymptotically approaches a Poisson distribution with mean $ke^{-k/n}$ where k is the number of requests attempting random access transmission each choosing one of n the available code-slots. Under this condition of "large" n, if we further approximate the model by ignoring the discrete nature of the framed structure, we can analyze the system by a continuous time Markov chain. Consider that the time for successful transmission and inter-arrival times for code requests are exponentially distributed with mean $1/\mu_k$ and $1/\lambda_k$, respectively, where is the total number of backlogged users and newly arrived requests. Then the state of the OC-ALOHA channel can be approximated [6] by a generalized M/M/1 queuing model with arrival rate $\lambda_k = \lambda$ and service rate $\mu_k = ke^{-k/n}$. Under M/M/1 formulation, we can obtain $p[\textit{the number of packets in systems} = k]$ as:

$$p_k = \frac{\lambda_{k-1}}{\mu_k} p_{k-1} = p_0 \left(\prod_{i=1}^{k} \frac{\lambda_{i-1}}{\mu_i} \right) \tag{2}$$

$$p_0 = \left[1 + \sum_{k=1}^{\infty} \prod_{i=1}^{k} \frac{\lambda_i}{\mu_i} \right]^{-1} \tag{3}$$

The mean delay time (MDT), $\bar{D}$ is given by:

$$\bar{D} = \left(\frac{1}{\lambda} \sum_{k=1}^{\infty} k p_k \right) - \frac{1}{2} \tag{4}$$

where 1/2 accounts for the average waiting frame time for new bandwidth requests due to frame synchronization which was not included in the discrete time models.

We consider next the FET. As defined in [12], FET is the average time for the system to make a first exit into the unstable region starting from an initially empty system, which means that all code slots in the frame are empty. In other words, the stability definition is as follows: a ranging channel is said to be stable if its service rate μ_k is greater than λ. Otherwise, the channel is said to be unstable.

Define k_{cr} to be the critical state of channel, as the minimum of k at which $\mu_k < \lambda$.

$$k_{cr} = \arg\min_{\mu_k < \lambda} k \tag{5}$$

Let the transition probabilities from state i to $i+1$ and from i state to $i-1$ be $\theta_i = \frac{\lambda_i}{\lambda_i + \mu_i}$ and $\bar{\theta}_i = 1 - \theta$, respectively. Then the mean transition time $t_{i,i+1}$ from state i to $i+1$ can be written as:

$$t_{i,i+1} = \frac{1}{\theta_i} \left[\frac{\theta_i}{\lambda} + \bar{\theta}_i \left(\frac{1}{\mu_i} + t_{i-1,i} \right) \right] \tag{6}$$

where $t_{01} = \frac{1}{\lambda}$. Using (6), we can express the FET $t_{0,k_{cr}}$ as:

$$t_{0,k_{cr}} = \sum_{i=0}^{k_{cr}-1} t_{i,i+1} \tag{7}$$

3.3 System Stability Evaluation

With the stability measure previously defined in Subsection 3.2, we demonstrate how the system can be evaluated by estimating the arrival rate and number of backlogged users to determine whether the stable state can be consistently maintained. This is achieved by estimating the critical state of the system, beyond which the system will go into the unstable state. We can obtain the estimated arrival rate using the first order autoregressive model as follows:

$$\tilde{\lambda}(i) = \theta_\lambda(i) + (1 - \theta_\lambda)\tilde{\lambda}(i-1) \tag{8}$$

where $ns(i)$, $\tilde{\lambda}(i)$ are the number of successes and the estimated arrival rate in the i^{th} frame, respectively, and θ_λ is the scalar smoothing factor in the arrival rate estimator (assuming $0 \leq \theta_\lambda \leq 1$).

Substituting $\tilde{\lambda}(i)$ into (5)-(7), we then obtain the estimated critical state, $\tilde{k}_{cr}$ and the estimated FET at the i^{th} frame, $\hat{F}(i)$ as follows:

$$\tilde{k}_{cr} = \arg\min_{\mu_k < \tilde{\lambda}} k \tag{9}$$

$$\hat{F}(i) = t_{0,\tilde{k}_{cr}} = \sum_{i=0}^{\tilde{k}_{cr}-1} t_{i,i-1} \tag{10}$$

In addition, the number of backlogged users can be estimated through the average number of access attempts until the ranging request is successful. When a SS sends a ranging channel transmission such as a bandwidth request to the BS using a randomly selected request code and the BS does not return any information to the SS, a code collision has occurred. Therefore, we assume that the SS reports the number of access attempts to the BS until the ranging request is successful. Thus, to estimate the number of backlogged users, we define the average number of access attempts until the ranging request is successful, $c(i)$ given by

$$c(i) = \frac{1}{ns(i)} \sum_{k=1}^{ns(i)} tx(k) \tag{11}$$

where $ns(i)$ and $tx(k)$ are the number of successful requests in the i_{th} frame and the number of transmission attempts of the k^{th} users, respectively.

Applying (9), we can estimate the number of backlogged users, $\tilde{b}(i)$, as:

$$\tilde{b}(i) = \theta_b c(i) + (1 - \theta_b)\tilde{b}(1-i) \tag{12}$$

where θ_b is a scalar smoothing factor in the estimator (assuming $0 \leq \theta_b \leq 1$).

Based on the estimated arrival rate ($\tilde{\lambda}(i)$), the estimated critical state ($\tilde{k}_{cr}$) in $\hat{F}(i)$ and the estimated number of backlogged users ($\hat{b}(i)$) from (8)-(12), we develop two novel algorithms for system stability.

Algorithm 1: Estimation-based Stabilization (ES) Algorithm with the Fixed Number of Ranging Code-Slots

1. At frame $i = 0$, set the number of code slots (n) for contention;
2. Estimate $\tilde{\lambda}(i)$, $\tilde{k}_{cr}$ in $\hat{F}(i)$ and $\hat{b}(i)$, in each ranging sub-frame using (8)-(12);
3. Each SS that has a ranging request to transmit obtains permission to transmit with probability $\gamma(i)$ and selects a code slot at random:

$$\gamma(i) = \tilde{k}_{cr}/(\hat{b}(i) + \tilde{k}_{cr}),\ (0 \leq \gamma(i) \leq 1)$$

4. Go to step (2) for time frame $i + 1$.

This algorithm estimates the permission probability ($\gamma(i)$) for transmitting ranging requests in frame $i + 1$ for the situation where a fixed number of code slots are available in the contention-based ranging channel per frame. System stability is maintained by controlling the permission probability $\gamma(\mathrm{i})$.

Algorithm 2: Estimation-based Stabilization (ES) Algorithm with Dynamic Ranging Code Allocation

$\delta(1) = 0$
n_{min}: the initiated number of code slots or the minimum number of code slots;

1. At frame $i = 1$, set the minimum number of code slots (n_{min}) for contention;
2. Estimate $\tilde{\lambda}(i)$, $\tilde{k}_{cr}$ in $\hat{F}(i)$ and $\hat{b}(i)$ in a ranging sub-frame i using(8)-(12) with

$$I_n(i){=}n_{min} + \delta(i);$$

   where $I_n(i)$ is the number of code slots which would dynamically be allocated by the variation parameter ($\delta(i)$) on the basis of n_{min}.
3. Calculate the permission probability ($\gamma(i)$) in this frame and the variation parameter ($\delta(i + 1)$) for the next frame.

$$\begin{aligned}\gamma(i) &= \tilde{k}_{cr}/(\hat{b}(i) + \tilde{k}_{cr});\ (0 \leq \gamma(i) \leq 1)\\ \delta(i + 1) &= I_n(i) - \lceil I_n(i) * (\gamma(i))\rceil\end{aligned}$$

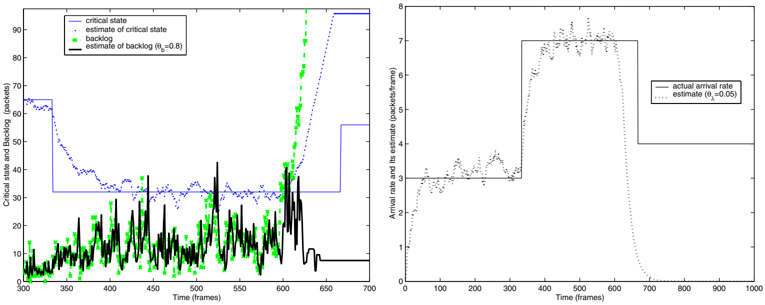
4. Go to step (2) for time frame $i + 1$.

This algorithm adjusts dynamically the number of code slots for ranging channel per frame based on the minimum number of code slots (n_{min}). After estimating $\tilde{k}_{cr}$ in $\hat{F}(i)$ and $b(i)$ in frame i, the BS calculates the permission probability ($\gamma(i)$) that indicates the degree of the instability of the system. The variation parameter ($\delta(i{+}1)$) is kept small if $\gamma(i)$ is close to one indicating a stable system; it is increased as $\gamma(i)$ gets smaller.

4 Numerical and Simulation Results

In this section, we present numerical and simulation results obtained using MATLAB. We assume that the arrival process of access requests at each SS is Poisson with rate λ, and let the number of backlogged requests vary according to the specified protocol. The duration of each simulation run is 1000 frames. Other basic assumptions are the same as given in Section 3.

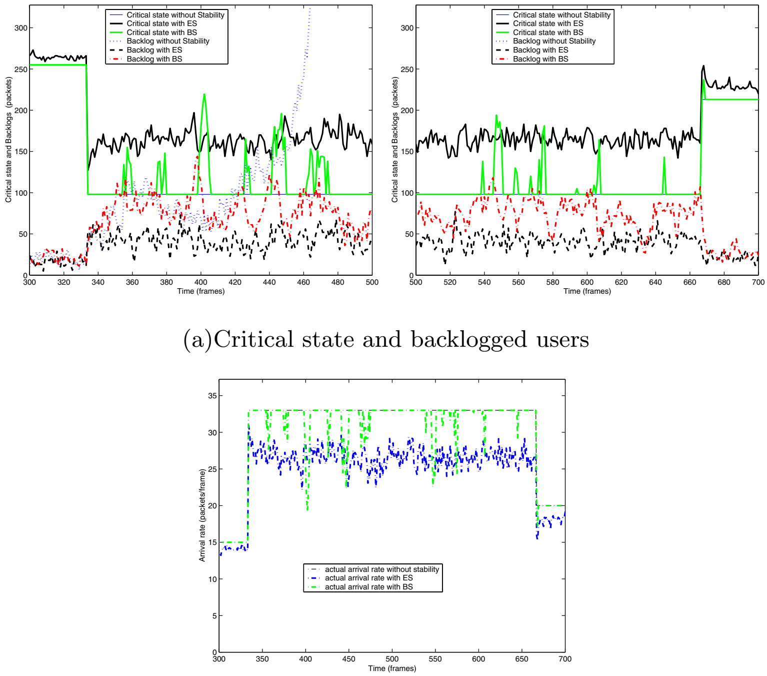
In [6], we observed that FET increases with the number of code-slots, n, and decreases as normalized arrival rate increases. This is because the number of collisions increases when n is reduced or the arrival rate is increased, resulting in a decrease in FET. Figs. 3(a) and (b) illustrate the trading relations among the estimated arrival rate, the estimated critical state, and the estimated number of backlogged users given by (8)-(10). The results have been obtained with n=21, θ_λ=0.05, θ_b=0.8, and the arrival rate is increased from 3 to 7 at around the 340th frame, and subsequently decreased to 4 at around the 670th frame. Fig. 3(b) shows that the estimated arrival rate quickly decreases to 0 at around the 650th frame. This is indicated by the point in Fig. 3(a) where the estimated number of backlogged users exceeds the estimated critical state as shown in around the 440th, the 525th and the 610th frames, which implies that the system has a high probability of falling into the unstable state. In this case, system stability can be maintained by implementing measures to reduce collisions, such as using a contention resolution algorithm or a mechanism to dynamically adjust the number of code-slots in the ranging sub-channel.



(a)Critical state and backlogged users (b)Actual arrival rate and its estimate

Fig. 3. Example of system stability evaluation

Figs. 4 and 5 illustrate the stability performance of algorithm 1 and 2, respectively, as presented in section 3. The results in Fig. 4 have been obtained using the following parameters for simulations: n=90, θ_λ=0.05, θ_b=0.8, the actual arrival rate is increased from 15 to 33, and subsequently decreased to 20. In Fig. 4(a), we can clearly observe that algorithm 1 always maintains the stable interval between the critical state and the number of backlogged users by

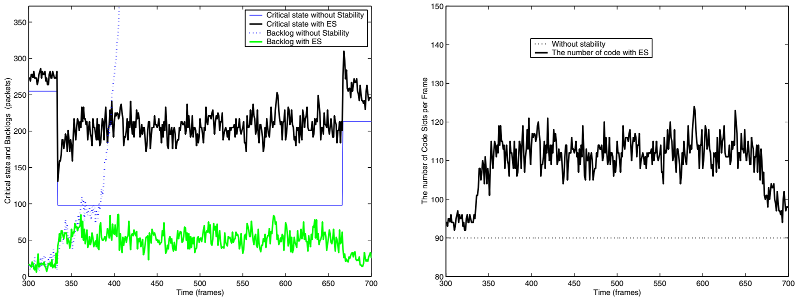


(a)Critical state and backlogged users

(b)Arrival rate

Fig. 4. Comparisons of performance of OC-ALOHA channel with a fixed number of code slots with (1) Estimation-based Stabilization (ES) Algorithm 1, (2) Pseudo-Bayesian Broadcasting Stabilization (BBS) and (3) no stabilization: (a) Critical state and backlogged users; (b) Arrival rate

strictly restricting the arrival rate (at the channel) even if the actual arrival rate (at the SSs) is quickly increased from 15 to 30 at around the 335^{th} frame as shown in Fig 4(b). By doing so, the numbers of backlog users at the same frame is increased from 20 to 60 while the critical state is decreased 260 to 125, and then they keep almost identical values with a little fluctuation when the frame proceeds until around the 665^{th} frame. On the contrary, the results for the Pseudo-Bayesian algorithm in Fig. 4 (b) show that the number of backlog users sometimes exceeds the critical state between the 335th frame and the 665^{th} when the arrival rate is high. Specifically, the number of backlogged users exceeds the critical state at around the 355^{th}, the 395^{th}, the 440^{th}, the 545^{th}, and the 605^{th} frames and then this unstable state is stabilized gradually at the following frames. These results show that our estimation-based stabilizing algorithm derived from the CTMC M/M/1 model using FET and number of backlogged users as the stability measures achieves better stabilization than the Pseudo-Bayesian Broadcasting algorithm.



(a)Critical state and backlogged users (b)The number of code slots per frame

Fig. 5. Performance of Algorithm 2 with the dynamic adjustment of number of code slots

To evaluate algorithm 2, we use the same simulation parameters as before, except that the number of code slots per frame is varied by estimating FET and the number of backlogged users. Figs. 5(a) shows that the system stability is always maintained even in the time interval that the system may fall into the unstable state as indicated by Fig. 4(b). Based on algorithm 2, the number of code slots per frame is adjusted upward from around 95 up to 120 and then downward to 100 when the actual arrival rate is increased from 15 to 33 and then decreased to 20.

In practice, the performance of our algorithms depends on the error performance of the physical layer given the particular spreading codes selected by the SS and the distribution of the power levels. Therefore, the evaluations can be extended to incorporate physical layer models. We leave these for future work.

5 Conclusions

In this paper, we have presented a new analytical technique based on the CTMC M/M/1 model for the contention-based CDMA ranging channel in IEEE 802.16a/d WMANs. This OC-ALOHA channel is inherently unstable. The model enables system measures such as the arrival rate, system backlog and critical to be easily estimated and utilized for system stabilization. We have proposed two novel stabilization algorithms, respectively for cases that the number of ranging code per frame is fixed or can be dynamically readjusted. In the former case, we have shown that system stability is always maintained by strictly restricting the ranging requests with permission probability ($\gamma(i)$). In the latter case, we have shown that the algorithm always maintains channel stability by dynamically allocating the number of code slots with permission probability ($\gamma(i)$) and the variation parameter ($\delta(i)$). Our novel approach of using a CTMC M/M/1 model to study system stability and to derive stabilization algorithms can be applied to other random access protocols and can be extended by incorporating more realistic physical layer models.

Acknowledgments

This work was supported by the Post-doctoral Fellowship Program of Korea Science & Engineering Foundation (KOSEF) and the Natural Sciences & Engineering Research Council of Canada through grant RGPIN 44286-04.

References

1. IEEE Standard for Local and Metropolitan Area Networks, "Part 16: Air Interface for Fixed Broad Wireless Access Systems - Amendment 2: Medium Access Control Modification and Additional Physical Layer Specification for 2-11GHz", IEEE Std 802.16aTM-2003.
2. Draft IEEE Standard for Local and metropolitan area networks, "Part 16: Air Interface for Broadband Wireless Access Systems", IEEE P802.16-REVd/D5-2004, May 2004.
3. http://www.wimaxforum.org/news/downloads/WiMAXWhitepaper.pdf
4. A. Brand and H. Aghavami, Multiple Access Protocols for Mobile Communications-GPRS,UMTS and Beyond, Wiley, 2002.
5. S. Simoens, P. Pellati, J. Gosteau and K. Gosse, "The Evolution of 5GHz WLAN toward Higher Throughputs", IEEE Wireless Communications, vol. 10, pp. 6-13, December 2003.
6. J.-J. Won, H.-H. Seo, C.-H. Cho, H.-W. Lee and V.C.M Leung, "Peformance Analysis of Contention-based Multiple Access Protocol in IEEE 802.16a Wireless MAN", Proc. 9th CDMA International Conference, pp. 334-338, Seoul, Korea, Oct. 2004.
7. S. Mangold, S. Choi, G. R. Hiertz, O. Klein, and B. Walke, "Analysis of IEEE 802.11e for QoS Support in Wireless LANs", IEEE Wireless Communications, vol. 10, pp. 40-50, Dec. 2003.
8. G.-H. Hwang and D.-H. Cho, "Adaptive random channel allocation scheme in HIPERLAN type 2", IEEE Communications Letters, vol. 6, pp. 40-42, Jan. 2002.
9. G. Anastasi and L. Lenzini, "HIPERLAN/1 MAC Protocol: Stability and Performance Analysis", IEEE J. Selected Areas in Commun., vol. 18, pp. 1787-1798, Sept. 2000.
10. D. Bertsekas and R. Gallger, Data Networks, Prentice Hall, 1992.
11. W. Feller, An Introduction to Probability Theory and Its applications, vol. 1, Cornell University, Jan. 1950.
12. L. Kleinrock and S. S. Lam, "Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation", IEEE Trans. Commun., vol. Com-23, pp. 410-423, Apr. 1975.
13. L. Kleinrock, Queuing Systems, vol. 1: Theory, Wiley-Interscience.

Load Balancing and Relaying Framework in TDD W-CDMA Multi-hop Cellular Networks†

Y. Hung Tam[1], Ahmed M. Safwat[2], and Hossam S. Hassanein[1]

[1] Telecommunications Research Laboratory, School of Computing,
Queen's University Kingston, ON, Canada K7L 3N6
{tam, hossam}@cs.queensu.ca
[2] The Laboratory for Advanced Wireless Networks,
Department of Electrical and Computer Engineering,
Queen's University, Kingston, ON, Canada K7L 3N6
ahmed.safwat@ece.queensu.ca

Abstract. In 3G cellular networks, high data-rates can be achieved. However, fundamental capacity limitation still exists. Call requests are frequently blocked in hotspot areas. Load balancing among cells helps to solve this problem and to utilize the radio resources. However, existing load balancing schemes are not designed specifically for 3G networks. Some schemes are not flexible enough or even practical for such networks. In this paper, we propose a novel load balancing and relaying framework for 3G TDD W-CDMA multi-hop cellular networks. This framework consists of a load balancing scheme called ALBA, a routing scheme called ACAR and a slot assignment scheme called E-DSSA. Our framework provides a realistic integrated load balancing solution for these networks. Simulation results show that this framework reduces the call blocking ratios of hotspot areas, balances load among cells and increases system throughput with low packet delay.

1 Introduction

Mobile communication has recently become affordable and popular. Wireless communications has gone through three generations. In 3G systems, CDMA is selected as the multiple access technology. A wider band is used to achieve higher data-rates. However, fundamental capacity limitation of these networks still exists. Call requests are frequently blocked in hotspot (congested) areas, such as city centers. Load balancing in a cellular network helps to solve this problem by shifting the load from a hotspot cell to less loaded cells. However, existing proposed load balancing schemes are not designed specifically for 3G networks such that some schemes are not practical or flexible enough for these networks. For examples, channel borrowing [3] and bandwidth migration [10] for load balancing works in conventional cellular networks, but

† This project has been supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) and Communications and Information Technology Ontario (CITO).

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1267 – 1280, 2005.

they are not practical for CDMA network because there are no additional channels (frequencies) or bandwidth that can be borrowed or migrated. In CDMA systems, one frequency is reused for all the cells in the systems. Cooperative (coverage) negotiation approach [5] varies the size and shape of cells using antenna technology to achieve load balancing. It is based on the assumption that cell capacity remains unchanged when cell size varies. This assumption does not apply to CDMA systems in which capacity of a cell decreases as coverage increases, i.e. cell breathing effect [7].

Recently, multi-hop relaying in cellular networks has gained attention because it has a potential to increase cell capacity and coverage. It also facilitates load balancing among cells. Load balancing schemes based on multi-hop relaying are Integrated Cellular and Ad Hoc Relay (iCAR) [4] and Pervasive Ad hoc Relaying for Cellular System (PARCelS) [16]. iCAR places low cost limited mobility Ad hoc Relay Stations (ARSs) in hotspot areas for traffic relaying and load balancing. However, iCAR is not flexible enough to handle the highly dynamic load situation of 3G cellular networks in which hotspots may surface at anytime, anywhere. The dynamic load situation is due to the fact that a large range of data-rates is provided for the users. PARCelS uses mobile nodes for relaying. When a base station (BS) is congested, mobile nodes search best routes to other non-congested cells. Route information is forwarded to BSs for selection. Obviously, the search requires high routing overhead.

Although multi-hop relaying concept is useful for load balancing, it raises routing and medium access issue. These issues could greatly affect load balancing. For Opportunity Driven Multiple Access (ODMA) [6] and Ad hoc-Cellular (A-Cell) relay [14] do not provide a routing solution. Multi-hop Cellular Network (MCN) [11], Hybrid Wireless Network (HWN) [2], Base-Centric Routing (BCR) [8] and PARCelS do not fully utilize the 3G CDMA systems for routing. In medium access, ISM bands and contention based medium access protocols are usually assumed for the relaying component. These introduce co-channel interference and medium access competition with other non-cellular users.

While a variety of load balancing and relaying strategies were proposed, none of them is perfectly fit into the 3G environment in terms of relaying, routing, and medium access. In this paper, we propose a novel load balancing and relaying framework for 3G TDD W-CDMA multi-hop cellular networks. This framework integrates load balancing, routing and channel allocation to provide a realistic load balancing solution for these networks. In the next section, we will describe the framework and its components. In section 3, the simulation model and simulation parameters will be presented. In section 4, simulation results and analysis of the results follows. In section 5, a conclusion and future work will be discussed.

2 A-Cell Load BAlancing and Relaying Framework (ALBAR)

We herein propose a novel load balancing and relaying framework called ALBAR for 3G TDD Wideband CDMA (W-CDMA) cellular systems. ALBAR is a centralized dynamic load balancing and relaying framework for consisting of three components: a load balancing scheme called ALBA, a load-aware routing scheme called ACAR, and a slot assignment scheme called E-DSSA. All three components are located at the Radio Network Controller (RNC) of a 3G WCDMA cellular network. This framework is designed based on the A-Cell relay architecture [14] in which directive antenna [12] is used. Figure 1 shows the architecture of the framework.

In this framework, ALBA monitors the traffic load condition. When load balancing is needed, ALBA performs load migration planning. ALBA interacts with ACAR to obtain valid route for relaying. ACAR interacts with E-DSSA for finding channels (time-slot code pair) for each node on the route. ACAR replies ALBA with the route finding. ALBA continues to perform load migration until the load in the system is balanced or no further load migration can be done. Then, ALBA sends signal through to the corresponding BSs for updating. The BSs send signals to the corresponding Source Nodes and Relaying Nodes for updating.

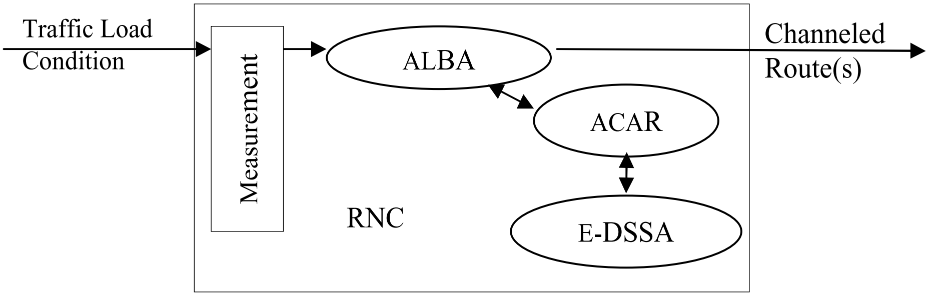


Fig. 1. ALBAR Framework Architecture

2.1 A-Cell Load Balancing (ALBA) Scheme

ALBA is a dynamic load balancing scheme for multi-hop ad hoc cellular system. The idea is to shift traffic load from a highly loaded cell to slightly loaded cells in a best effort manner. This manner is used because although source nodes are available for load migration, relaying route may not exist. ALBA is not only suitable for hotspots environment, but also for any heterogeneous load environment. Figure 2 shows the ALBA scheme.

ALBA periodically checks the load status of the networks (Line A1). If Network Load Deviation (D_N) is greater than Network Load Deviation Threshold (D_{NThres}), ALBA starts load migration planning. ALBA selects a Target Cell, a Source Cell, and a Source Node of the Source Cell (Line B2 to B5). The Target Cell is the least load cell in the network. The Source Cell is a cell neighboring the Target Cell with the highest load above the Target Cell's load. To proceed further, the Neighboring Load Deviation (d_n) between the Source Cell and the Target Cell must be greater than the Neighboring Load Deviation Threshold (d_{nThres}) and the Call Blocking Ratio (*CBR*) of the Source Cell also needs to be greater than or equal to the *CBR* of the Target Cell; otherwise, no load will be migrated (line B4). Next, ALBA selects a Source Node based on its Migration Priority (*MP*) level calculated as follows.

$$\text{Migration Priority } MP_i = k_1 * C_{ij} + k_2 * f(d_i) \quad (1)$$

$$\text{Where} \quad f(d_i) = \begin{cases} 4 & | \; d_i =< R \\ 3 & | \; R < d_i =< 1.25\,R \\ 2 & | \; 1.25\,R < d_i =< 1.5\,R \\ 1 & | \; 1.5\,R < d_i \end{cases}$$

Where MP_i is the Migration Priority level of a Connection j of Source Node i; C_{ij} is the QoS class of the connection; d_i is the distance between the Source Node and the BS of Target Cell; $f(d_i)$ is a mapping function to map the distance d_i to a distance factor which is an integer from 1 to 4; and R is the cell radius. Constant k_1 and k_2 are weighing factors for C_i and $f(d_i)$, respectively. The values (1 to 4) of the distance factor are chosen such that the distance factor could break even with the value of quality of service class at some location.

Once a Source Node is selected, ALBA calls ACAR to find a Channeled Route for relaying (Line B6). ACAR replies ALBA with the result of route finding. Whether the result is a success or not, ALBA continues to select another Source Node for load migration until the load of the Source Cell is not greater than the load of the Target Cell or all Source Nodes in the Source Cell are tried (line B7).

If no load can be migrated between the two Cells, ALBA selects the next highest load neighboring cell, higher than the load of Target cell, as the new Source Cell (Line B8). If load migration still fails, ALBA continues to try the next highest load neighboring cell until all neighboring cells are tried or load migration is a success (Line B8). If load migration still fails, ALBA selects the next least load cell as the new Target Cell. If load migration is still a failure, ALBA continues to find a new Target Cell until all cells are tried or load migration is a success.

If load migration between the Source Cell and the Target Cell is a success, ALBA reviews the new load distribution of the network. If the D_N is still greater than D_{NThres}, ALBA selects another set of Target Cell, Source Cell and Source Node for load migration (Line B10). These procedures continue until D_N is less than D_{NThres} or no further load migrations can be done, i.e. all cells are tried. Then, ALBA sends signals through the RNC to the corresponding BSs to update their Channel Allocation Tables and Channels Pools. BSs send signals to the nodes on both old and new routes to update their Channel Tables and Routing Tables.

Illustration of ALBA Scheme

Figure 3 illustrates how ALBA balances loads in a heterogeneous traffic load environment. First, we choose a Target Cell. Cell C is chosen because it is one least load cell. Then, we choose the Source Cell. Cell A is chosen because it is the highest load neighboring cell of the Target cell C. After three load units are migrated from cell A to cell C, the new load situation is reviewed (see Figure 3b). Load migration continues. Figure 3g represents the balanced load situation.

ALBA Scheme

A) Load Monitoring State

```
Checks the load status of the network through RNC in every Load Sampling Period ($T_s$)
If Network Load Deviation ($D_N$) > $D_{NThres}$, then
    Transit to Load Balancing State
```

B) Load Balancing State

```
Do
    Do select * (next) least load cell as Target Cell ($C_{Trg}$)
        Do select † (next) highest load (higher than $C_{Trg}$) non-tried neighboring cell of the
        $C_{Trg}$ as Source Cell ($C_{Src}$ )
            If Neighboring Load Deviation > $d_{nThres}$ and $CBR$ of $C_{Src}$ >= $CBR$ of $C_{Trg}$,
                Do select non-tried Src with highest MP for neighboring load migration.
                    Perform ACAR to find a Channeled Route from the Src to the BS of
                    $C_{Trg}$.
                While (load of $C_{Src}$ > load of $C_{Trg}$ and there are still non-tried Src)
        While (neighboring load migration fails and there are still non-tried $C_{Src}$)
    While (neighboring load migration fails and there are still non-tried $C_{Trg}$)
While ($D_N$ > $D_{NThres}$ and there are still non-tried $C_{Trg}$)
ALBA sends signals to corresponding BS(s) through RNC to update their Channel
    Allocation Tables and Channels Pools.
BS(s) sends signals to the nodes on both old and new route(s) to update their Channel
    Table(s) and Routing Table(s). (Order of signal sending can be simultaneously or least
    loaded cell first.)
Transit to Load Monitoring State.
```

* - If several least load cells are tied, randomly select one which is not an adjacent cell of previous C_{Trg} if possible.

† - If several highest load cells are tied, randomly select one of these cells which is not a previous C_{Trg} if possible.

Definition:

- **Source Node** (***Src***) is a mobile node that generates traffic.
- **Channeled Route** is a route (or path) on which each node can be assigned a Channel.
- **Channel Pool** is a pool of available channels in the BS.
- **Channel Table** stores channel information for the connections of Source Nodes.
- **Channel Allocation Table** stores channel information for the connections in the BS.
- **Neighboring Load Deviation** (d_n) is the load difference between two neighboring cells.
- **Neighboring Load Deviation Threshold** (d_{nThres}) is a value above which load migration between neighboring cells can proceed.
- **Network Load Deviation** (D_N) is the load difference between the highest load cell and the least load cell in the network.
- **Network Load Deviation Threshold** (D_{NThres}) is a value for triggering load migration.
- **Load Sampling Period** (T_s) is the time interval for checking load of the network.
- **Call Blocking Ratio** (*CBR*) is the ratio of number of calls blocked to the number of calls requested.

Fig. 2. ALBA Scheme

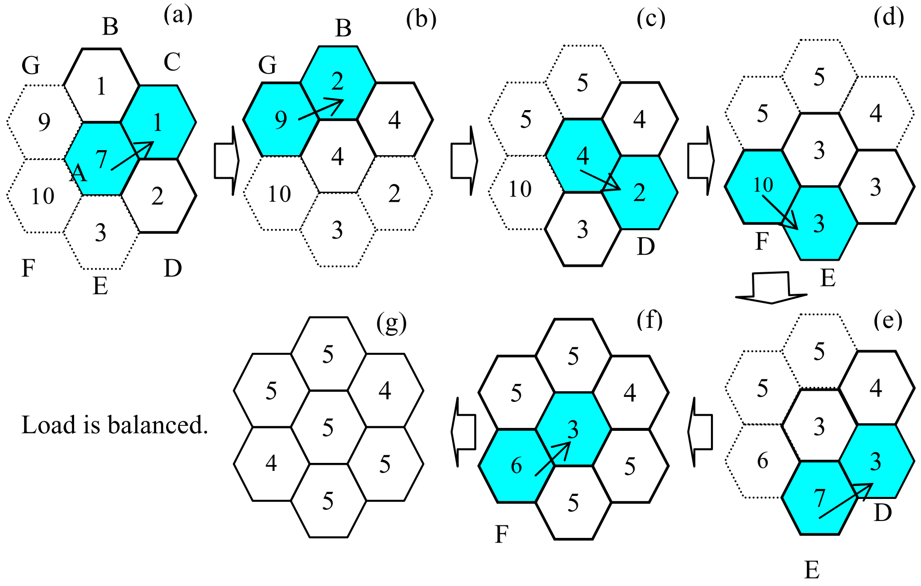


Fig. 3. Load balancing of ALBA in heterogeneous load environment

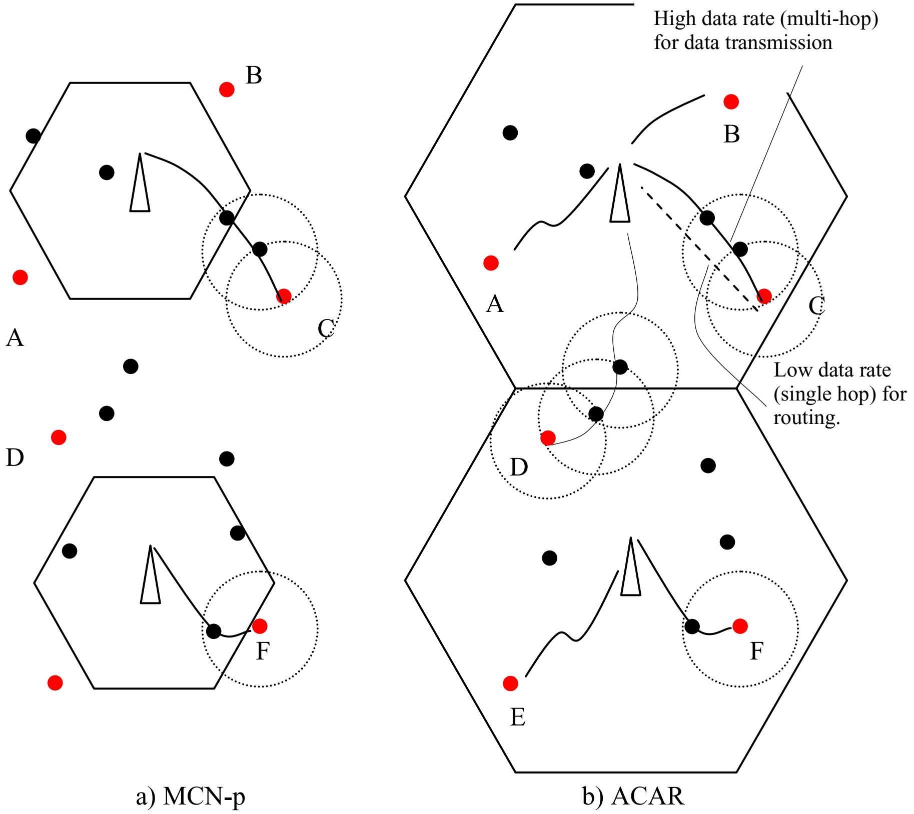
2.2 A-Cell Adaptive Routing (ACAR) Scheme

ACAR is a centralized on-demand load-aware routing scheme specifically designed for 3G CDMA cellular systems. ACAR has two mechanisms: Routing Discovery and Route Maintenance. The idea of ACAR is to make use of the cell size flexibility, i.e. the cell breathing effect, of a CDMA cellular system. Route discovery and route maintenance can be done in a single hop with long range and low data-rate while data communication can be done in a multi-hop with short range and high data-rate fashion. In this way, routing overhead can greatly be reduced and the benefits of multi-hop relaying remain. In addition, no potential call requests, which are within the maximum coverage of a cell, will be denied. Figure 4 illustrates this by comparing MCN-p [11] and ACAR. In MCN-p, since the cell size is shrunk, potential calls from node A, B, D, and E becomes unreachable. In ACAR, these nodes still reach the BS. Thus, no potential call is denied. In Figure 3b, node B and node E are using single hop routing. Node C and F are using multi-hop routing. Node D is using inter-cell routing for load balancing. Details of the scheme can be found in [15].

2.3 Extended-Delay Sensitive Slot Assignment (E-DSSA) Scheme

E-DSSA is a slot assignment scheme which is an extension of Delay Sensitive Slot Assignment (DSSA) [1]. DSSA is a slot assignment scheme which is based on A-Cell relay architecture in which directive antennas and Global Positioning System (GPS) is used. In addition to the merits of spatial reuse, channel conflict resolution and the low packet delay attained by DSSA, E-DSSA avoids consecutive channel conflict and

Last Hop Node Interference. The former is used to avoid signal interference that a mobile should not send and receive data on the same channel at the same time. The latter is used to avoid assigning a channel to a node (current node) such that the Last Hop Nodes on other routes is transmitting on that channel towards the next hop node of that current node. This may happen because the Last Hop Nodes may transmit using Normal Transmission Range (R_{normal}) instead of Relaying Transmission Range. Details of the scheme are in [15].


B
A
C
D
F
High data rate (multi-hop) for data transmission
B
A
C
Low data rate (single hop) for routing.
D
F
E

a) MCN-p

b) ACAR

Fig. 4. Comparison of the routing behavior in MCN-p and ACAR

Illustration of E-DSSA Scheme

In Figure 5, assume node C and B are successfully assigned the channels (slot 5, code 16) and (slot 4, code 2) respectively. Node A is proposed to be assigned a channel (slot 3, code 1). To avoid channel conflict, node x should not be receiving on this channel and node z should not be transmitting on this channel; otherwise, channel conflict occurs. In consecutive conflict avoidance, assume a new call request from node D arrives and path D-B-C-BS is the shortest path. Channel (slot 4, code 2) cannot be assigned to node C for this new path because node C is currently receiving on this channel for the connection of node A. In Last Hop Node Interference avoidance, when a new call request from node E arrives, node E can transmit using R_{normal} because it is the Last Hop Node. Node E cannot be assigned a channel that node C is using for reception; otherwise, a channel conflict at node C occurs. In this case, the channel is (slot 4, code 2).

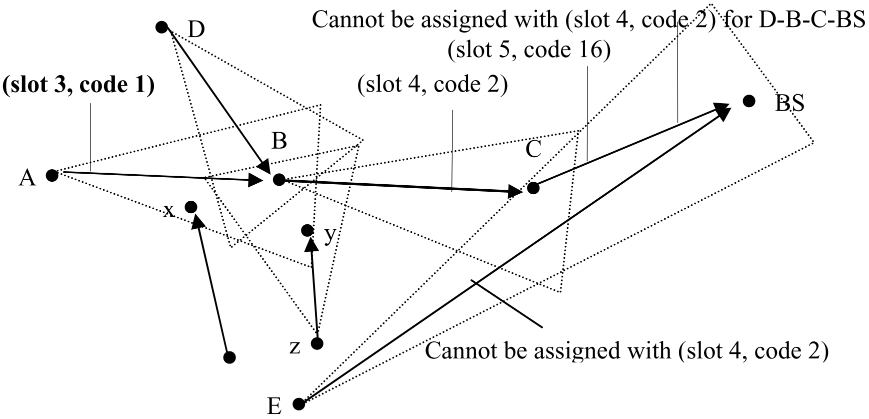

D
Cannot be assigned with (slot 4, code 2) for D-B-C-BS
(slot 5, code 16)
(slot 3, code 1)
(slot 4, code 2)
BS
B
C
A
x
y
z
Cannot be assigned with (slot 4, code 2)
E

Fig. 5. Conflict and interference resolution of E-DSSA

3 Simulation Model

Our simulation model is a three-cell one (see Figure 6). This three-cell model is a generalized model that can be applied to a 3-tier cell system or any geographically limited situation. In fact, this model is a stricter version of a 3-tier cell model because the hot cell has only one neighboring cell for load migration.

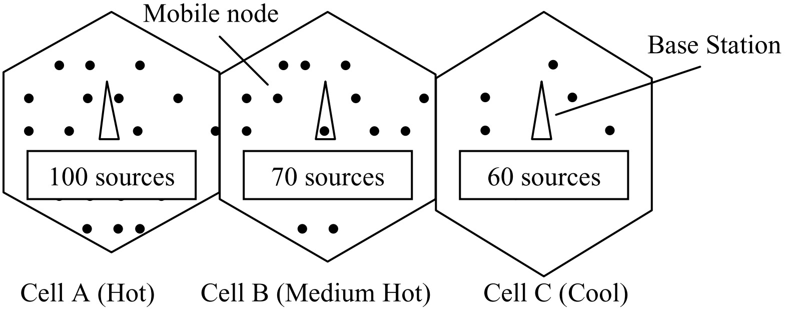


Fig. 6. Three-cell model

Experimental Setting and Parameters

In this model, cells A, B and C are respectively hot, medium hot and cool. The numbers of source nodes in cells A, B, and C are respectively 100, 70, and 60. All three cells are connected to the Radio Network Controller (RNC). ALBA, ACAR, and E-DSSA are located at the RNC. Each source node generates call requests at an average rate of 0.1 calls/second following Poisson distribution. The average duration of each call is 5 seconds with exponential distribution. The transmission rates for relaying and without relaying are set at 1Mbps. The data rate of each connection is 13.8kbps. Simulation duration is 30 seconds. During the simulation, when a source node places a call request, the call is admitted or rejected by a call admission control (CAC) mechanism. If the call is admitted, CAC calls ACAR to find a channeled route for the connection. The connection can be single or multi-hop depends on the availability of reliable relaying nodes. ACAR chooses a reliable relaying node for relaying based on their state information such as nodal speed, current load and remaining battery capacity. If ALBA is enabled, ALBA checks the load condition of the three cells every 10 seconds. If Network Load Deviation is greater than one connection, load balancing proceeds.

Since the load balancing through relaying is affected by the location of the relaying nodes, we vary the number of relaying nodes in the hot cell or the medium hot cell to investigate the load balancing effect. Two scenarios are formulated as below.

Scenario 1: Relaying nodes in cell A:B:C = (0/ 40/ 80/ 120/ 160):160:120
Scenario 2: Relaying nodes in cell A:B:C = 160: (0/ 40/ 80/ 120/ 160): 120

For scenario 1, while the numbers of relaying nodes in cell B and cell C are fixed at 160 and 120 respectively, the number of relaying nodes in cell A is varied from 0 to 160 at every step 40. Scenario 2 is similar to scenario 1 except that we vary the number of relaying nodes on cell B instead while the number of relaying nodes in cell A and cell C are fixed at 160 and 120 respectively. The performance metrics used are call blocking ratio of the three cells, load deviation among cells, system throughput, and end-to-end delay of packets under load balancing and without load balancing situation. The OPNET [13] commercial modeling and simulation tool is used. For this simulation, we assume perfect power control, perfect physical medium and slow mobility. We also assume that each relaying node has enough battery capacity. The complete parameter description can be found in [15].

4 Simulation Results

The followings are parts of the result of our work. More results are in [15].

Call Blocking Ratio (CBR)

Figure 7 (scenario 1) shows that, without load balancing, CBR of cell A is approximately 19 to 23%. When the load balancing is enabled, the CBR of cell A drops deeply. The sudden drop, even though there is no Relaying Nodes in cell A, is because there are already many Relaying Nodes in cell B. Once load balancing is triggered, the load of the Source Nodes in cell A which are near to the border of cell B can be relayed immediately to cell B. Then, more capacity is available in cell A.

Hence, the CBR of cell A decreases. As cell B receives load from cell A, cell B's capacity decreases and its CBR increases. This effect also applies in cell C. This demonstrates the chain reaction of the load balancing effect. When the number of relaying nodes in cell A increases, the CBR of cell A further decreases because more Relaying Nodes in cell A increases the chance for the Source Nodes in cell A in getting routes to cell B.

Figure 8 (scenario 2) shows a similar result except that there is no sudden drop of CBR even though there are many Relaying Nodes in cell A. The reason is when there is no or not enough Relaying Nodes in Cell B, there is no or not enough relaying route for load migration. Thus, the reduction in CBR in cell A is relatively smaller. So, the presence of Relaying Nodes in the neighboring cell (cell B) of a hot cell (cell A) provides a faster reaction for load balancing.

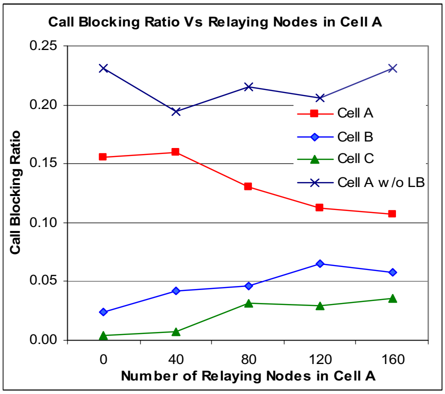


Fig. 7. Call Blocking Ratio among cells with and without load balancing (scenario 1)

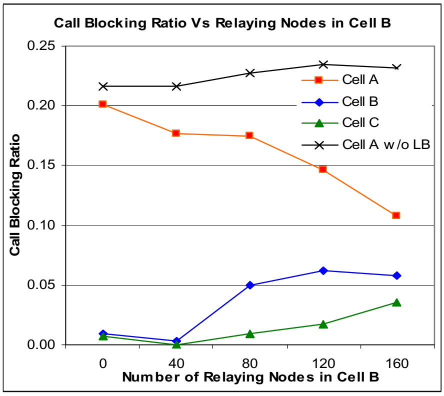


Fig. 8. Call blocking Ratio among cells with and without load balancing (scenario 2)

Load Deviation

Figure 9 and Figure 10 show that load deviation is reduced in both scenarios when load balancing is enabled. The maximum load deviation, which is the load difference between the highest and the least load cells, follows a similar trend. When the number of Relaying Nodes in cell A or cell B increases, the load deviation is further reduced. This is because more routes are available for load migration. The figures also show that the load deviation drops faster when the number of Relaying Nodes is beyond 40. The reason is when the number of relaying nodes is sufficiently large, i.e. beyond 40, there is a higher chance that the relaying nodes fall near to the border of cell B for relaying. In Figure 10, when the number of Relay Nodes in cell B is beyond 120, the rate of reduction slows down because most Source Nodes has already shifted their load to their neighboring cell.

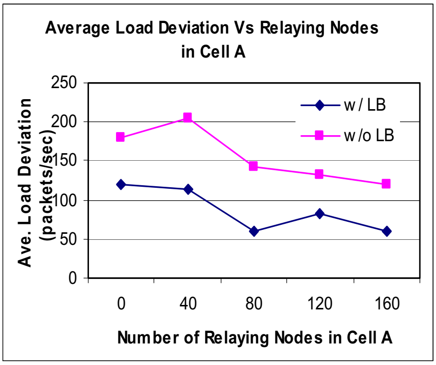


Fig. 9. Average load deviation with and without load balancing (scenario 1)

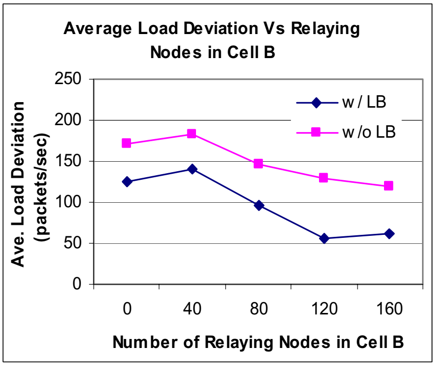


Fig. 10. Average load deviation with and without load balancing (scenario 2)

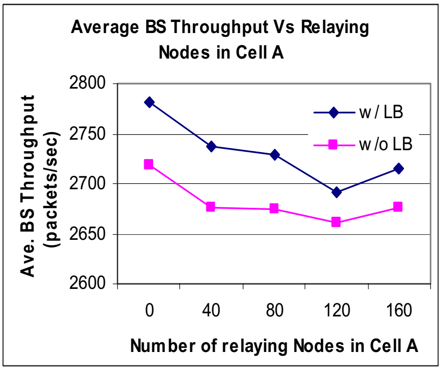
Throughput

Both Figure 11 and Figure 12 show that the average BS throughput is increased when load balancing is enabled. Since the load balancing effect reduces call blocking ratio, more calls can be served. Thus, the throughput is increased. Figure 11 shows that the throughput decreases gently when the number of Relaying Nodes increases. This is because the number of hops per route increases. This introduces higher packet delay. Fewer packets arrive at their BSs in same period of time. Thus, throughput is relatively lower. In Figure 12, the throughput does not decrease with the increase in the number of Relaying Nodes in cell B. This is because the Relaying Nodes that are already in cell A dominate the packets' delay. Source Nodes in cell A communicates with the BS using multi-hop relaying from the beginning, increasing the packet delay. Hence, the initial throughput in Figure 12 is lower than the initial throughout in Figure 11.

End-to-End Delay

Figure 13 shows that the average end-to-end delay of packets is higher when load balancing is enabled. This is because the average number of hops in inter-cell routing is higher than the average number of hops in intra-cell routing. When the number of Relaying Nodes in cell A increases, the delay in both load balancing and without load balancing situations increases. This is because the increase in the number of Relaying Nodes in cell A increases the number of hops in both intra-cell and inter-cell routing. When the number is more than 120, the increase of delay slows down because most routes reach the maximum hop count that has been set. Figure 14 shows a similar result except the delay is lower with load balancing when the number of Relaying Nodes in cell B is around zero. This is because a large number of Relaying Nodes are already present in cell A. The Source Nodes in cell A communicates with the BS of cell A using multi- hopping which causes high packet delay. Therefore, the average delay in cell A dominates the average delay of the system. Thus, a higher delay is obtained even though load balancing is not enabled. When load balancing is enabled, traffic of Source Nodes near to the boundary of cell A starts to be relayed to cell B.

Since there are fewer Relaying Nodes in cell B at that time, the number of hops in the inter-cell routing is smaller. In other words, the Source Nodes, which are originally using more hops to communicate with their BS, uses fewer hops to communicate with their new BS. Thus, the delay is smaller. When the number of Relaying Nodes in cell B is beyond 20, the delay with load balancing enabled is higher than the delay without load balancing. This is consistent with the result in Figure 13.



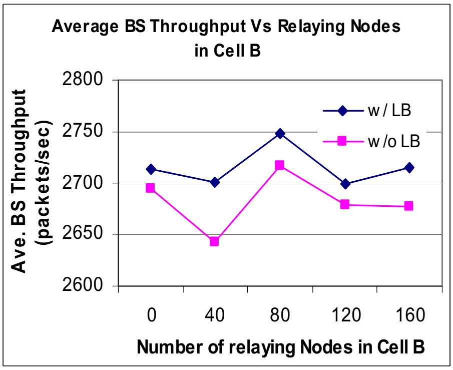
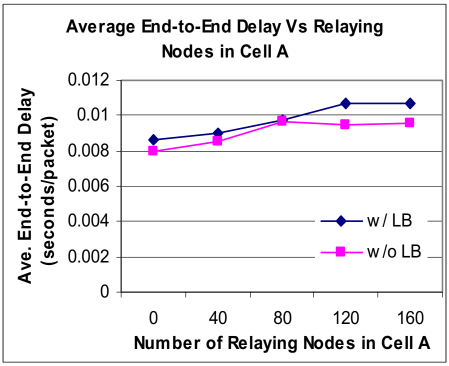


Fig. 11. Average Throughput with and with out load balancing (scenario 1)

Fig. 12. Average Throughput with and without load balancing (scenario 2)



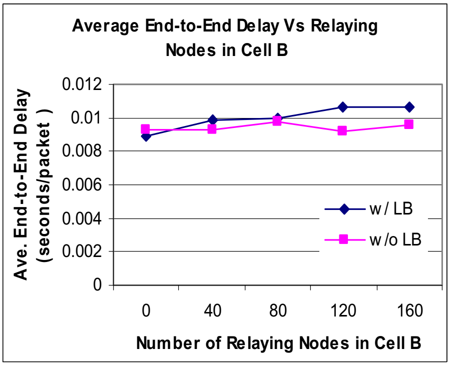


Fig. 13. Average End to End Delay with and without load balancing (scenario 1)

Fig. 14. Average End to End delay with and without load balancing (scenario 2)

5 Conclusions and Future Work

A-Cell Load Balancing Relaying Framework (ALBAR) integrates load balancing, routing and channel allocation functionalities to provide a realistic load balancing solution for the 3G TDD W-CDMA multi-hop cellular environment. This framework can be applied to different environments in terms of network topology, load distributions, and relaying route availability. In particular, ALBA can be applied to any heterogeneous load environment. ACAR fully utilizes the dynamic cell size characteristic of CDMA cellular systems such that routing overhead can greatly be reduced and the benefits of multi-hop relaying remain. E-DSSA inherits the merits of DSSA and addresses the dynamic cell size characteristic of CDMA cellular system. Simulation results show that the framework reduces the call blocking ratio of a hotspot cell significantly. Load deviation among cells and maximum load deviation in the network are also reduced. System throughput increases. End-to-end delay of packet increases only slightly. The results also show that the presence of Relaying Nodes in the neighboring cell of a hot cell provides a faster reaction for load balancing.

In the future, optimization of the components of the framework can be performed. A Quality of Service (QoS) provisioning component can also be added to this framework. A detailed comparison of ACAR with other multi-hop cellular routing protocols can be performed. The applicability of this framework to other networks such as mobile ad hoc networks and sensor networks is also an interesting area for study.

References

1. Al-Riyami, M., Safwat, A. M., Hassanein, H. S.: A Slot Assignment Scheme for TDD W-CDMA Multi-hop Cellular Networks. Proceedings of the 22nd Biennial Symposium on Communications, Queen’s University, Canada, May (2004) 235-237
2. Chang, R., Chen, W., Wen, Y.: Hybrid Wireless Network Protocols. IEEE Trans. Vehicular Technology, vol. 52. July (2003) 1099-1109
3. Das, S. K., Sen, A.K., Jayaram, R.: A Structured Channel Borrowing Scheme for Dynamic Load Balancing in Cellular Networks. IEEE ICDCS (1997)
4. De, S., Tonguz, O., Wu, H., Qiao, C.: Integrated Cellular and Ad hoc Relay (iCAR) Systems: Pushing the Performance Limits of Conventional Wireless Networks. 35th Annual Hawaii International Conference on System Sciences (HICSS) (2002)
5. Din, L., Bigham, J., Cuthbert, L., Nahi, P., Parini, C.: Intelligent Cellular Network Load Balancing Using Cooperative Negotiation Approach. IEEE WCNC (2003)
6. 3rd Generation Partnership Project (3GPP); Technical Specification Group Radio Access Network; Opportunity Driven Multiple Access. Sophia Antipolis (1999) (3G TR 25.925)
7. Holma, H., Toskala, A.: WCDMA for UMTS, Radio Access for Third Generation Mobile Communications. Revised edition, John Wiley & Sons (2001)
8. Hsu, Y.C., Lin, Y.D.: Base-Centric Routing Protocol for Multi-hop Cellular Networks. IEEE GLOBECOM (2002)
9. Johnson, D. B., Maltz, D. A.: Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile Computing, Kluwer Academic Publishers (1996)
10. Kim, S., Varshney, P. K.: Adaptive Load Balancing with Preemption for Multimedia Cellular Networks. IEEE WCNC (2003) 1680-1684
11. Lin, Y. D., Hsu, Y.C.: Multihop Cellular: A New Architecture for Wireless Communications. IEEE INFOCOM, Mar. (2000)
12. Ohira T., Gyoda, K.: Design of electronically steerable passive array radiator (ESPAR) antennas. Antennas and Propagation Society International Symposium (2000)
13. OPNET Technologies Inc., www.opnet.com , Sept. (2004)

14. Safwat, A.: A-Cell: A Novel Multi-hop Architecture for 4G and 4G+ Wireless Networks. Proceedings of IEEE Vehicular Technology Conference (VTC) (2003)
15. Tam, Y. H.: A Load balancing and Relaying Framework in A-Cell Networks. Master thesis, Queen's University, Canada, Sept. (2004)
16. Zhou, J., Yang, Y.: PARCelS: Pervasive Ad-hoc Relaying for Cellular Systems. Proceedings of Med-Hoc-Net, Sardegna, Italy, Sept. (2002)

Stochastic Decision-Based Analysis of Admission Control Policy in Multimedia Wireless Networks

Nidal Nasser

Department of Computing & Information Science,
University of Guelph, Guelph, Ontario, N1G 2W1 Canada
nasser@cis.uoguelph.ca

Abstract. Providing multimedia services with Quality of Service (QoS) guarantees in next generation wireless cellular networks poses great challenges due to the scarce radio bandwidth. Effective Call Admission Control (CAC) is important for the efficient utilization of the limited bandwidth. In this paper, the design of CAC for a decision-based Markovian model of multimedia wireless networks is considered. We propose an optimal CAC policy that aims at maximizing the system utilization while stratifying the QoS constraints bounding the handoff dropping probability of each traffic class. We used the Semi-Markov Decision Process (SMDP) technique to represent the proposed CAC. The optimal CAC decisions for each state are found by solving the linear programming formulation problem. Numerical and Simulation results show that the system upholds the handoff call dropping probability required by each traffic class while maximizing revenue for service providers. The requirements of the mobile users are hence satisfied in periods with different loads, including overload. Moreover, the implemented policy ensures efficient utilization of resources. This latter facet is highly desirable by service providers.

1 Introduction

With the explosive growth of the wireless communication market, the demand for developing multimedia applications is increasing rapidly. Next generation high-speed Wireless Cellular Networks (WCNs) are expected to support multimedia applications (audio phone, video-on-demand, video-conferencing, Web services, etc.). The third generation (3G) - Universal Mobile Telecommunication System (UMTS) is an example of such a network. Multimedia applications necessitate support of different classes of traffic with diverse Quality of Service (QoS), which need to be guaranteed by WCN. To achieve this goal, QoS provisioning is critical.

One of the main design issues in WCNs is the implementation of Call Admission Control (CAC) policy. The function of the CAC policy is defined as a set of actions to determine if the call request can be accepted or rejected. The condition for accepting a new call request is the availability of sufficient resources (e.g. wireless link bandwidth) to guarantee the QoS requirements without adversely affecting existing calls.

For WCNs, two types of QoS parameters have been introduced which are: New Call Blocking Probability (NCBP) and Handoff Call Dropping Probability (HCDP)

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1281 – 1296, 2005.

[1]. The NCBP is the percentage of new calls blocked. New call blocking occurs when all the channels of the wireless medium are busy upon a new call request. The HCDP is the percentage of blocked handoff calls. When a user moves from one cell to a neighboring cell, a handoff takes place. The user requests a channel from the new base station, and if all the channels are busy dropping takes place. Since call dropping of established connections is usually more annoying than rejection of a new connection request, it is widely believed that a wireless network must give higher priority to the handoff connection requests as compared to new connection requests. Therefore, we focus on the handoff dropping probability as the main QoS requirement.

Several CAC policies have been proposed [2], [3], [4] and applied to support identical traffic connections such as guard-channel policy [2]. This policy deals with handoff calls. It reserves a subset of bandwidth allocated to a given cell for handoff calls. Clearly, increasing the number of guard channels will reduce the HCDP and, at the same time, it may increase the NCBP. Berqia et al. [5] has shown that the guard-channel policy is optimal for minimizing a linear objective function of the new and handoff calls. In [6], a learning automaton is used to find the optimal number of guard channels. However, both models assume that the traffic of all connections is identical. This assumption, however, is not valid if multimedia services are to be supported, since multimedia connections may differ in the amount of bandwidth they need to meet their QoS requirements. In order to achieve these requirements, many CAC policies have been presented to support the multimedia services in WCNs [7], [8], [9]. These, however, do not consider the problem of optimizing the admission policy to minimize the probability of dropping a handoff call. In [10], [11], [12], the theory of Semi-Markov Decision Processes (SMDP) is used to construct optimal CAC policies. Yoon et al. [10] proposed a CAC policy using SMPD and used an approximation technique to solve the SMDP-formulated problem. Xiao et al. [11] proposed a genetic algorithm to obtain a near-optimal CAC for multimedia wireless networks and showed some numerical results to illustrate the concept. In [12], Bartolini et al. applied a SMDP formulation to analyze a CAC policy for multiple classes of traffic. A call may transit from one class to another to satisfy the admission requirement. An approximation with iterative solution has been used to obtain the most optimal CAC policy. However, iterative solutions are computationally infeasible for practical size networks. The common disadvantage of the above SMDP models is that they all provide sub-optimal solutions.

In this paper, the theory of SMDP is used to construct an optimal CAC policy for WCNs that support multimedia services. The policy extends and generalizes the guard-channel policy in that it reserves an appreciate amount of bandwidth for different types of classes to assure QoS of handoff calls. We call the modified policy Multi-Class Guard Channel Call Admission Control (MCGC-CAC). Optimal decisions of the MCGC-CAC are obtained by applying SMDP linear programming formulation under hard constraint of the handoff call dropping probability. Our approach is able to achieve optimal performance, maximum bandwidth utilization and guarantee that all handoff calls of different classes of traffic with diverse QoS requirements will experience a dropping probability less than the designated maximum value for each service class.

The rest of this paper is organized as follows. The network model is presented in Section 2. The multi-class guard channel CAC policy that supports multimedia traffic is proposed in Section 3. The SMDP framework and the optimal CAC policy are presented in Section 4. Numerical results are shown in Section 5. Finally, conclusions drawn from the paper are discussed in Section 6.

2 Network Model Description

We consider a network model that supports multiple classes of adaptive multimedia services, which have diverse bandwidth and QoS requirements. We first describe our system model, and then we describe the traffic model.

2.1 Cell Configuration

Our approach is based on decomposing the wireless cellular network into individual sub-systems, each corresponding to a single cell. The correlation between these sub-systems (models), results from handoff connections between corresponding cells. Therefore, we can model the system at a single-cell level.

Suppose there are K classes of calls in the system (cell). The bandwidth of a class-i, i.e., the number of channels required to accommodate the call, is given by b_i. Let the total bandwidth units (in number of channels) in each cell be the same and denoted by B. The classes are indexed in an increasing order according to their bandwidth requirements, such that:

$$b_1 \leq \ldots \leq b_i \leq b_{i+1} \leq \ldots \leq b_K$$

In this paper, we assume a Fixed Channel Allocation (FCA) environment. Furthermore, it is assumed that the bandwidth of wideband calls can be scattered across the bandwidth pool as in [9].

2.2 Traffic Model

New call arrivals and handoff call arrivals of class-i (i = 1, 2, …, K) connections are assumed to follow a Poisson process with rates λ_{nc_i} and λ_{h_i}, respectively. The total arrival rate of class-i connections is $\lambda_i = \lambda_{nc_i} + \lambda_{h_i}$. The Call Holding Time (CHT) of a class-i call is assumed to follow an exponential distribution with mean $1/\mu_{b_i}$.

For mobility characterization, we assume the following simple model. The Cell Residence Time (CRT), i.e., the amount of time during which a mobile terminal stays in a cell during a single visit, is assumed to follow an exponential distribution with mean $1/h$ [13]. We assume that the CRT is independent of the service class; hence, connections in any class follow the same CRT distribution. Note that the parameter h represents the call handoff rate.

The channel holding time is the minimum of the CHT and the CRT. As the minimum of two exponentially distributed random variables is also exponentially distributed, then the channel holding time for class-i connection is, therefore, assumed to be exponentially distributed with mean $1/\mu_i$ where $\mu_i = \mu_{b_i} + h$. Our traffic model is illustrated in Fig. 1.

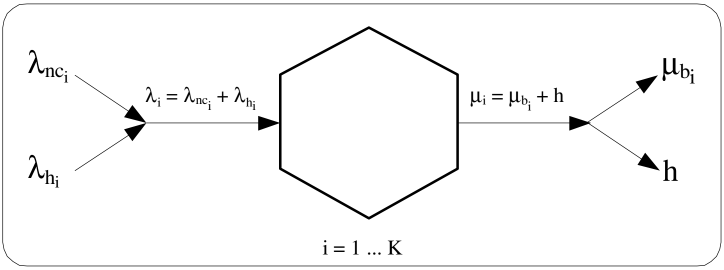


Fig. 1. Traffic model in wireless cellular network

3 Multi-class Guard Channel Policy

The proposed Call Admission Control (CAC) policy, Multi-Class Guard Channel (MCGC), is a generalized and enhanced version of the guard-channel policy [2]. The guard-channel policy is modified to be applicable for different traffic classes with different QoS requirements. The principle idea of the existing guard-channel policy is as follows. Considering a single cell with a fixed amount of bandwidth capacity B bandwidth units (in a number of channels), the guard-channel policy gives a higher priority to handoff call requests as opposed to new call requests by reserving a portion of the bandwidth (channels) resource for handoff calls. More specifically, a new call request is admitted only when there are less than B_{Th} channels occupied, where B_{Th} is a threshold between 0 and B. on the other hand, a handoff request is rejected only when all B channels are occupied. As a result, $B_R = B - B_{Th}$ channels are the guard channels, which are used only by handoff calls.

In this work we generalize the principle idea of the guard-channel policy for the case in which K separate classes of traffic with diverse bandwidth and QoS requirements exist in the system. Therefore, we design a multi-class guard channel policy that aims to achieve two goals: call prioritization and class differentiation. Similar to the guard-channel policy, our policy reserves a portion of the total bandwidth for handoff calls, thus giving them priority over new calls and potentially providing them with lower handoff call dropping probability. On the other hand, the policy achieves class differentiation as follows. Calls with lower bandwidth requirements will have a better chance at occupying the bandwidth than those with higher bandwidth requirements.

Consider a system with the following parameters:

- B_{Th} is the total available bandwidth for different classes of new calls and handoff calls.
- $B_R = B - B_{Th}$ is the reserved bandwidth for different classes of handoff calls.
- x_i is the number of ongoing class-i connections, i = 1, …, K.
- b_i is the requested bandwidth of a connection of class-i.
- $b_{i,assigned_j}$ is the assigned bandwidth for call j, $1 \leq j \leq x_i$, of class-i.

The modified CAC policy can be described as follows:

```
Upon the arrival of a new call of class-i, $b_i$
{
    if ($b_i + \sum_{i=1}^{K}\sum_{j=1}^{x_i} b_{i,assigned_j} \le B_{Th}$)
         then accept the call, and $x_i = x_i + 1$
    else block the call
 }
Upon the arrival of a handoff call of class-i, $b_i$
{
    if ($b_i + \sum_{i=1}^{K}\sum_{j=1}^{x_i} b_{i,assigned_j} \le B$)
         then accept the call, and $x_i = x_i + 1$
    else drop the call
 }
Upon the completion of a call of class-i or handoff to another cell
{
    $x_i = x_i - 1$
 }
```

That is, a newly arriving call of class-i will be granted admission if its requested bandwidth, b_i, plus the current total bandwidth of ongoing connections (new calls and handoff calls) for all classes are less than or equal to B_{Th}. Whereas, A handoff arriving call of class-i will be granted admission if its requested bandwidth, b_i, plus the current total bandwidth of ongoing connections (new calls and handoff calls) for all classes are less than or equal to the total capacity B.

4 SMDP Formulation

The aim of this section is to formulate the Multi-Class Guard Channel (MCGC) policy as a Semi-Markov Decision Process (SMDP). Before we proceed an overview of SMDP is given.

4.1 Overview of SMDP

The semi-Markov decision process is a stochastic process that describes the evolution of dynamic systems controlled by sequences of decisions or actions. The dynamic system at random discrete time points (epochs) is observed and classified into one of a finite number of states. After classification, one of a finite number of possible actions must be chosen and the corresponding revenue for each state is gained due to this decision [14]. For each state *x*, a set of actions is available. If the system is in state *x* and action *a* is chosen then:

1. The next state, *y*, of the system is chosen according to the transition probability p_{xy}^{a}.
2. The time until the transition from *x* to *y* occurs is τ(x, a).

After the transition occurs, an action is again chosen and (1) and (2) are repeated. Markovian properties are satisfied if at a decision epoch the action *a* is chosen in the current state *x*, and the state at the next decision epoch depends only on the current state *x* and the chosen action *a*. They are thus independent of the past history of the system.

4.2 SMDP Formulation for the Multi-class Guard Channel Policy

In this section, we first formulate the MCGC policy in a cell as a SMDP. Then the SMDP-formulated problem is represented as a Linear Programming (LP) problem, which aims at both maximizing the bandwidth system utilization while stratifying the QoS constraint to upper bound of the handoff dropping probability.

Whenever there is an arrival of class–i call (new or handoff), CAC must make some decisions. A decision is defined as an action of the system that takes place in a particular state upon a call arrival. The decisions include whether the system accept the call or not. The CAC policy selects its decision from a finite decision (action) space. The possible decision for all kinds of traffic (new call or handoff call) is {accept/reject}.

The SMDP formulation is characterized by the following 5 ingredients:

1- State Space:
The current state of the system (cell) can be represented by the vector:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_K). \tag{1}$$

The non-negative integer x_i denotes the number of ongoing new calls and handoff calls class-i connections. Let Ω denote the state space required to represent our CAC policy. Also, let $\Omega_{B_{Th}} \subset \Omega$ denote the set of states at which a class-i new call and handoff call are accepted, and let $\Omega_{B_R} \subset \Omega$ denote the set of states at which a class-i handoff call is only accepted. Thus, according to the multi-class guard channel policy:

$$\Omega_{B_{Th}} = \{\, \mathbf{x} \in \Omega \mid 0 \le \sum_{i=1}^{K} x_i b_i \le B_{Th};\ \mathrm{x_i} \ge 0,\ 0 \le B_{Th} \le B \,\}, \text{ and} \tag{2}$$

$$\Omega_{B_R} = \{\, \mathbf{x} \in \Omega \mid B_{Th} < \sum_{i=1}^{K} x_i b_i \le B;\ \mathrm{x_i} \ge 0 \} \tag{3}$$

Therefore, the state space for the multi-class guard channel policy that contains all admissible states can be represented by:

$$\Omega = \Omega_{B_{Th}} \cup \Omega_{B_R} \tag{4}$$

A Markov state diagram for the guard-channel policy is shown in Fig. 2 where B = 6, $B_R = 3$, $b_1 = 1$ and $b_2 = 2$.

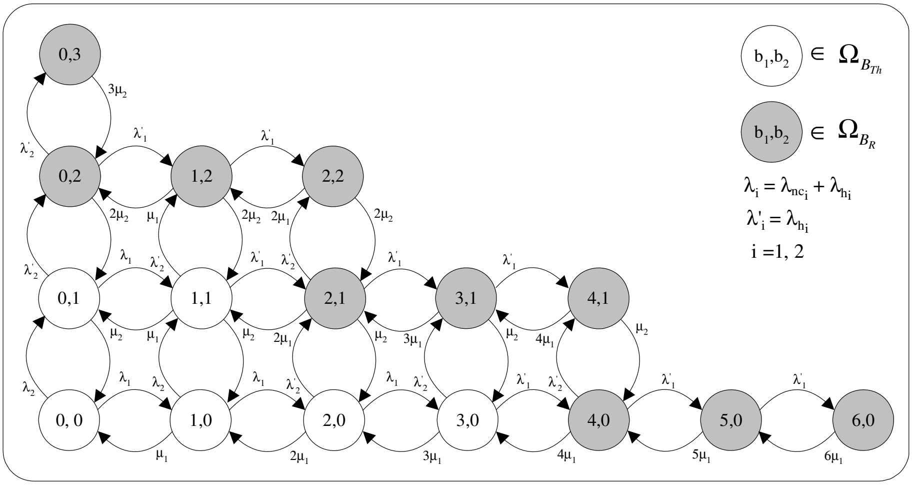


Fig. 2. Markov state diagram for the guard-channel policy with B = 6, $B_R = 3$, $b_1 = 1$ and $b_2 = 2$

At each state *x*, the CAC policy should select an accept/reject decision for all kind of traffic arrivals, new calls or handoff calls. The set of all the selected decisions are called the admission policy of the system.

2- Decision Epochs and Action Space:

When an arriving new call or handoff call desires to be admitted into the system, the CAC policy will make a decision as to whether or not to grant admission. We define the decision epochs as the times immediately following the arrival events. A decision epoch is a vector $v = (x, e)$, where x is the vector of class-i calls as in (1) and $e \in \{a_{nc_1}, \ldots, a_{nc_K}, a_{h_1}, \ldots, a_{h_K}\}$. The variable e represents the event type of an arrival and the indicators a_{nc_i}, and $a_{h_i} \in \{0, 1\}$ for i = 1, …, K, denote the origination of class-i new call within the cell and the arrival of a class-i call due to handoff from an adjacent cell respectively.

When the system is in state *x*, an accept/reject decision must be made for each type of possible arrival, i.e., an origination of a class-i new call, or the arrival of a class-i handoff call. Thus, the action space A can be expressed as follows:

$$A = \{a = (a_{nc_1}, \ldots, a_{nc_K}, a_{h_1}, \ldots, a_{h_K}) : a_{nc_i}, a_{h_i} \in (0, 1), i = 1, \ldots, K\}, \tag{5}$$

where $a_{nc_i} = 0$ (or 1): reject (or accept) the new call of class-i, and

$a_{h_i} = 0$ (or 1): reject (or accept) the handoff call of class-i.

For example, when K=2 and $(a_{nc_1}, a_{nc_2}, a_{h_1}, a_{h_2}) = (0, 1, 1, 1)$ this indicates that only new calls of class-1 will be rejected by the CAC policy.

For a given state $x \in \Omega$, the admissible action space for the system, $A_x \subset A$, is defined as follows:

$$A_x = \{a \in A: a_{nc_i} = 0 \text{ and } a_{h_i} = 0 \text{ if } y = x + e_i \notin \Omega, i = 1, \ldots, K\}. \tag{6}$$

Under this action space, a class-i call is accepted when the state is *x* if and only if $y = x + e_i \in \Omega$. In other words, A_x is composed of all those actions in A that do not result in a transition to a state $y = x + e_i \notin \Omega$. Here e_i is a vector of zeros, except for the ith component which it is 1.

3- Expected Sojourn Time:

If the system is in state $x \in \Omega$ and the action $a \in A_x$ is chosen, then the expected sojourn time of the state *x*, τ(x, a), is given by:

$$\tau(x, a) = \left[\sum_{i=1}^{K}(x_i\mu_i + \lambda_{nc_i}a_{nc_i} + \lambda_{h_i}a_{h_i})\right]^{-1}, \tag{7}$$

where $x_i\mu_i$ represents the rate at which calls terminate within the cell, $\lambda_{nc_i}a_{nc_i}$ represents the rate at which new calls originate from the cell, and $\lambda_{h_i}a_{h_i}$ represents the rate of incoming handoffs from adjacent cells to this cell.

4- State Dynamics:

The state dynamics of a SMDP is completely specified by stating the transition probabilities between the system states. Let p_{xy}^{a} be the transition probability that at the next decision epoch the system will be in state *y* if the present state is *x* and the action *a* is chosen, where $a \in A_x$. For $y \in \Omega$, we have the following cases:

$$\begin{array}{lll} \textit{Class-i arrival:} & y = x + e_i: & p_{xy}^{a} = (\lambda_{nc_i}a_{nc_i} + \lambda_{h_i}a_{h_i})\tau(x,a) \\ \textit{Class-i departure:} & y = x - e_i: & p_{xy}^{a} = (x_i\mu_i)\tau(x,a) \\ \textit{Otherwise:} & & p_{xy}^{a} = 0, \qquad \forall i \in \{1,\ldots,K\},\ x \in \Omega \end{array} \tag{8}$$

5- A Revenue Function:

Let r(x, a) be the revenue rate when the cell is in state *x* and action *a* is chosen. If r_i is the revenue rate of class-i, the total revenue rate for the system is calculated by:

$$r(x,a) = \sum_{i=1}^{K} r_i x_i \cdot \tag{9}$$

Assuming that revenue is given by the number of channels assigned, the total revenue rate in state *x* is equal to the system utilization in state *x* as follows:

$$r(x,a)=\sum_{i=1}^{K} b_i x_i , \tag{10}$$

where r_i is replaced by b_i in (9).

4.3 Constructing Optimal Mutli-class Guard Channel Policy

SMDPs are usually analyzed and solved within the framework of discrete-time average cost Markov decision processes; see [14] for a detailed discussion. We define decision variable π_{xa} , $x \in \Omega$ and $a \in$ A$_x$, as the long-run fraction of time at which the state x chooses action a, and the set of π_{xa} collectively determines the CAC policy. Searching for the optimal CAC policy is equivalent finding those decision variables for all states. This can be done by solving the following SMDP Linear Programming (LP) formulation which aims to maximize long-run system utilization and to guarantee QoS for handoff calls.

$$\text{Maximize} \sum_{x\in\Omega}\sum_{a\in A_x} r(x,a)\tau(x,a)\pi_{xa} \tag{11}$$

$$\text{Subject to:} \sum_{x\in\Omega}\sum_{a\in A_x} \tau(x,a)\pi_{xa} = 1 \tag{12}$$

$$\sum_{a\in A_x}\pi_{ya} = \sum_{x\in\Omega}\sum_{a\in A_x} p_{xy}^{a}\pi_{xa} \ , \ y\in\Omega \tag{13}$$

$$\pi_{xa} \geq 0, x\in\Omega, a\in A_x \tag{14}$$

The term $\tau(x,a)\pi_{xa}$ in (11) can be interpreted as the long-run fraction of decision epochs when the system in state x and action a is chosen. Hence, the objective function (11) is the system utilization. The constraints are: the normalization condition (12), the global balance equation from the long-run viewpoint (13), and (14) is the constraint on decision variables. The optimal feasible solutions π_{xa}^{*} to equation (11) give the optimal CAC policy.

A nice feature of the LP formulation for solving SMDP is that it permits optimization over additional constraints. Since dropping handoff calls is usually less desirable and less tolerable than blocking newly initiated calls, we focus on the handoff dropping probability as the main QoS requirement. Thus, we consider the QoS requirements of the upper bound on the handoff dropping probability of each class-i. Let $d(x,a)=1-a_{h_i}$. The stationary handoff call dropping probability for class-i when action a is selected is given by [19]:

$$P_{d_i} = \sum_{x\in\Omega}\sum_{a\in A_x} d(x,a)\tau(x,a)\pi_{xa}, i=1,\ldots,K\ . \tag{15}$$

We therefore add the QoS constraint of the handoff dropping probability to equation (11) to limit the handoff call dropping probabilities for class-i below a target value $P_{QoS\,i}$:

$$P_{d_i} \leq P_{QoS\,i}, i = 1,..., K\,, \tag{16}$$

where P_{QoS_i} denotes the maximum allowable handoff dropping probability of a class-i.

5 Numerical Results

In this section, the performance of the Multi-Class Guard Channel (MCGC-CAC) policy is evaluated using simulation and compared with the Upper-limit (UL) CAC policy [16]. Before proceeding with evaluating the performance of the MCGC-CAC policy, we first describe the simulation model that is used in this paper. Then, we show that the MCGC policy actually achieves its goals in terms of call prioritization and class differentiation. Finally, we show how our formulation to the MCGC-CAC policy as SMDP achieves the optimal solution and outperforms the UL CAC policy, in which it maximizes the bandwidth utilization of the system (cell) and at the same time guarantees QoS for handoff multimedia traffic.

5.1 Simulation Model

We simulate one cell with a diameter of 1 km (i.e., micro cellular environment). The base station resides at the center of the cell. Part of the BS functionalities is the Admission Controller (AC) that operates the CAC policy. The AC components are

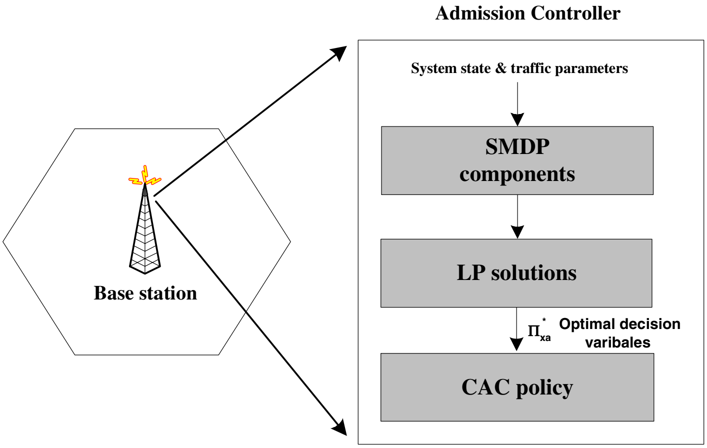

Admission Controller
System state & traffic parameters
SMDP components
LP solutions
Π^*_{xa} Optimal decision varibales
CAC policy
Base station

Fig. 3. Admission controller module

shown in Fig. 3 and operate as follows. Given the current state of the system and the traffic parameters, SMDP components are calculated. Then, the AC uses the LP technique to compute the optimal decision value, π^*_{xa} , that aims at maximizing the system utilization function as given in (11). Solving the LP problem to find the optimal call admission decisions is an offline procedure. That is, the decisions are obtained before invoking the CAC mechanism.

We develop our simulation model based on the following assumptions.

- The total bandwidth capacity of the simulated cell is B basic bandwidth units.
- Two classes of multimedia service are considered. Bandwidth requirement of class-i connection is b_i (i = 1, 2). Class-2 has a higher priority than class-1.
- Since class-2 traffic has higher priority than class-1, we assume that the maximum allowable handoff dropping probability for class-2 calls, P_{QoS_2} , always has a higher value than P_{QoS_1} .
- New call requests of class-i are generated in the cell according to a Poisson process with rate λ_{nc_i} (calls/second). A newly generated call can randomly appear at any position in the cell with an equal probability. The handoff call arrival rate of class-i is assumed to be proportional to the new call arrival rate of class-i by $\lambda_{h_i} = (h/\mu_{b_i})\lambda_{nc_i}$ [17] for i = 1, 2. Thus, the total arrival rate of class-i calls is $\lambda_i = \lambda_{nc_i} + \lambda_{h_i}$.
- We assume the call holding time of a class-i is exponentially distributed with mean $\mu_{b_i}^{-1}$ (seconds). Also, we assume the cell residence time is exponentially distributed with mean h^{-1} (seconds). The channel holding time of class-i calls is exponentially distributed with mean μ_i^{-1} $(\mu_i = \mu_{b_i} + h)$.
- Mobiles can travel in one of eight directions with equal probability. A constant randomly selected speed is assigned to a mobile when it enters a cell either at call initiation or after handoff. The speed is obtained from a uniform probability distribution function ranging from V_{min} to V_{max}.

The simulation model is very flexible in which all the above parameters are provided as an input to the simulation program. Thus, this will allow us to test the system with different scenarios. In this paper, we limit our experimental tests to the simulation parameters values that are shown in Table 1. However, we believe that the higher bandwidth capacity, the more efficiency our policy can achieve since solving the LP problem to find the optimal call admission decisions is an offline procedure. That is, the decisions are found out before the operating of CAC. In addition, techniques for solving large-scale LP problems such as [15], [18] can be applied to the cases of large cellular systems and/or cellular systems with larger capacity. Also, even though there may be some discrepancy between the real bandwidth values of the multimedia service and the values in Table 1, we believe that our experiments can reflect the real system's behavior.

The performance measures obtained through simulation are New Call Blocking Probability (NCBP), Handoff Call Dropping Probability (HCDP) and bandwidth utilization.

Table 1. Simulation parameters

Parameter	Value	Unit
B	30	bbu
b_1	1	bbu
b_2	2	bbu
$1/\mu_{b1} = 1/\mu_{b2}$	500	sec
1/h	100	sec
V_{min}	10	km/hr
V_{max}	60	km/hr

5.2 Performance Evaluation – Relaxed Constraints

To explore the comprehensive effect of our proposed policy, MCGC, on the NCBP and the HCDP for each traffic class, handoff QoS constraints are relaxed. That is, upper bound of handoff dropping probability of class-i is set to one, i.e., P_{QoS_1} = P_{QoS_2} = 1. In this experiment, we use the bandwidth threshold, B_{Th} = 20 bandwidth units (channels).

Fig. 4 shows the effect of varying Erlang load on the NCBP and the HCDP of each traffic class. From this figure, we observe that the new call blocking probabilities and the handoff dropping probabilities of both classes increase as the Erlang load increases. However, the HCDP is always lower than the NCBP as a result of the 10 bandwidth units ($B_R = B - B_{Th}$) reserved exclusively for handoff calls. Moreover, the NCBP and HCDP of class-1 connections are lower than those of class-2 connections.

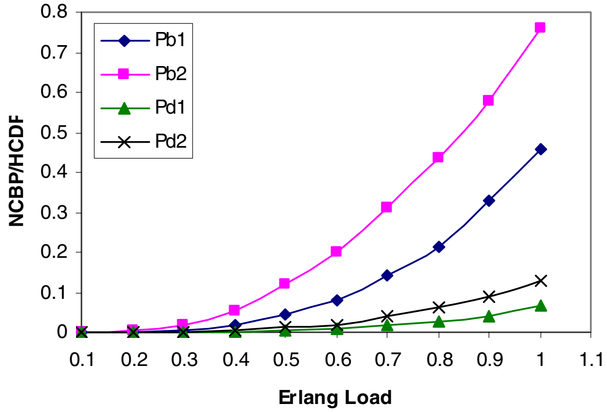


Fig. 4. Effect of varying the Erlang Load on the NCBP and the HCDP

This is because class-1 connections have lower bandwidth requirements, and therefore, they have a better chance in occupying the bandwidth which results in a higher blocking/dropping probability for the higher bandwidth class. The above observations show the ability of our policy to prioritize between new calls and handoff calls and to differentiate between traffic classes.

5.3 Performance Evaluation – Hard Constraints

In this section, we design several experiments to compare our proposed policy to the UL CAC policy with respect to the HCDP and the bandwidth utilization. The UL CAC policy blocks a new call request of class-i if the number of the calls is greater or equal to an upper-limit value, i.e. threshold t_i. The UL CAC policy used for comparison has a threshold $t_1 = 10$ and $t_2 = 15$. Simulation parameters are the same as in Table 1 Section 5.1 and $P_{QoS_1} = 0.02$ and $P_{QoS_2} = 0.04$.

Fig. 5 shows the HCDP for both policies as Erlang load increases. It is shown that the HCDP for the MCGC-CAC policy is bounded by 0.02 and 0.04 for class-1 and class-2 connections, respectively, and therefore, their QoS requirements are satisfied. On the other hand, the UL CAC policy cannot guarantee such bound, especially when the Erlang load increases.

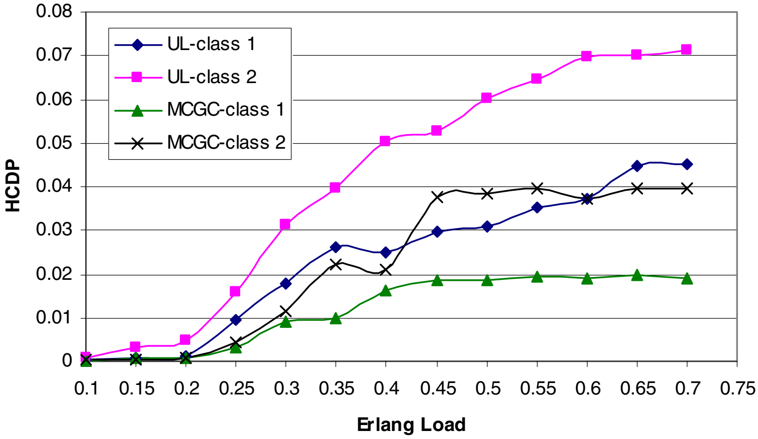


Fig. 5. Handoff call dropping probability vs. Erlang Load

Different cases were investigated as shown in Table 2 to compare between the two policies in term of the bandwidth utilization. The bandwidth utilization of both policies is obtained while the guard channel B_R is varying. The last column of Table 2 shows the utilization improvement ratio, UIR, of our policy over the UL CAC policy. The UIR is obtained as follows:

$UIR = \frac{U_{MCGC} - U_{UL}}{U_{UL}} * 100$, where U_{MCGC} is the utilization of the multi-class guard channel policy and U_{UL} is the utilization of the upper-limit policy.

Table 2. Utilization improvement results

No.	P_{QoS_1}	P_{QoS_2}	B_R	U_{MCGC}%	U_{UL}%	UIR%
1	0.02	0.04	5	**29.08**	**26.88**	*8.18*
2	0.02	0.04	10	**28.10**	**26.88**	*4.52*
3	0.02	0.04	15	**27.01**	**26.88**	*0.48*

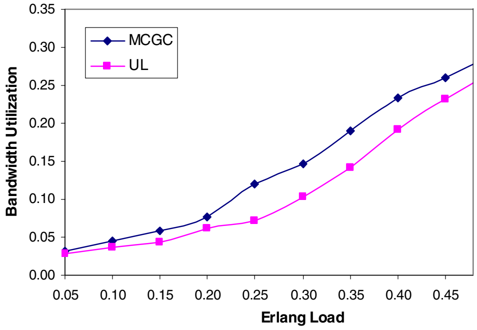


Fig. 6. Bandwidth Utilization vs. Erlang Load (P_{QoS_1}= 2%, P_{QoS_2} = 4%, B_R = 5, t_1 = 10, t_2 = 15)

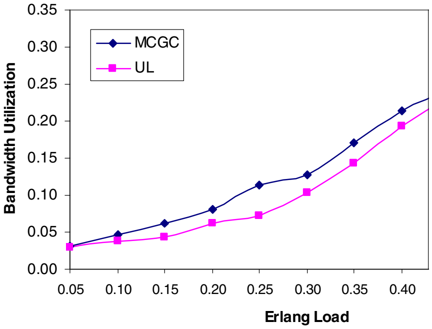


Fig. 7. Bandwidth Utilization vs. Erlang Load (P_{QoS_1}= 2%, P_{QoS_2} = 4%, B_R = 10, t_1 = 10, t_2 = 15)

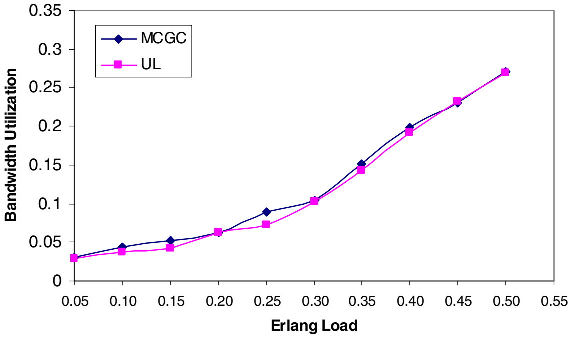


Fig. 8. Bandwidth Utilization vs. Erlang Load (P_{QoS_1}= 2%, P_{QoS_2} = 4%, B_R = 15, t_1 = 10, t_2 = 15)

Fig. 6-8 demonstrate the effect of varying the Erlang load on the bandwidth utilization for both policies considering all the cases of Table 2. It is observed that the MCGC-CAC policy (MCGC in the Figures) over SMDP always gets better utilization than the UL CAC policy. Also, we observe that as the reserved bandwidth for handoff calls, $B_R = B - B_{Th}$ increases, the bandwidth utilization decreases. This is because we are reserving more bandwidth for future handoff calls so that the resource is less utilized. However, the upper bound for the handoff dropping probability is guaranteed which indicates a clear conflict relation between the bandwidth utilization and QoS.

6 Conclusions

Providing multimedia services with Quality of Service (QoS) guarantees in next generation high-speed wireless cellular networks poses great challenges due to the scarce radio bandwidth. Effective Call Admission Control (CAC) is important for the efficient utilization of the limited bandwidth. In this paper, we generalized and enhanced the well-known guard-channel policy to accommodate multimedia traffic. Under our new policy, we reserve a fraction of the total available bandwidth in a cell exclusively for multiple classes of handoff calls with each class having distinctively different QoS requirements. The multi-class guard channel CAC policy is formulated as a Semi-Markov Decision Process (SMDP) with constraints on the dropping probabilities of multimedia handoff calls. SMDPs, in the generic sense, are a proven method for evaluation of CAC policies and QoS parameters in wireless networks. In this work, the optimal multi-class guard channel policy decisions are obtained by applying SMDP linear programming formulation. The optimal CAC decisions for each state are found by solving the linear programming formulation with the objectives of maximizing the system utilization and guaranteeing QoS of handoff calls. Simulation results show that the multi-class guard channel policy over SMDP outperforms the upper-limit CAC policy while maximizing the bandwidth utilization and satisfying the upper bound dropping probability of each class of handoff calls.

References

1. S. S. Rappaport, "The Multiple-Call Hand-off Problem in High-Capacity Cellular Communications Systems", *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 3, August 1991, pp. 546-557.
2. D. Hong and S. S. Rappaport, "Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and Non-prioritized Handoffs Procedures", *IEEE Transactions on Vehicular Technology*, Vol. 35, No. 3, Aug. 1986, pp. 77-92.
3. S. Choi and K. Shin, "A Comparative Study of Bandwidth Reservation and Admission Control Scheme in QoS-sensitive Cellular Networks", *Wireless Networks*, Vol.6, 2000, pp.289–305.
4. Y. Iraqi and R. Boutaba, "An Adaptive Distributed Call Admission Control for QoS-sensitive Wireless Mobile Networks", *Proceedings of the IEEE Wireless Communications and Networking Conference*, Chicago, IL, Sept. 2000, pp.449–453.
5. A. Berqia and N. Mikou, "Model and Optimal Call Admission Policy in Cellular Mobile Networks", *Springer Lecture Notes in Computer Science*, LNCS 1815, October 12, 2001, pp. 920.

6. H. Beigy and M. Meybodi, “A Learning Automata Based Dynamic Guard Channel Scheme”, Springer *Lecture Notes in Computer Science, LNCS 2510*, October 10, 2002, pp. 643.
7. K. Ross and D. Tsang, “The Stochastic Knapsack Problem”, *IEEE Transactions on Communications*, Vol. 37, No. 7, July 1989, pp. 740-747.
8. S. Jordan and P. Varaiya, “Control of Multiple Service, Multiple Resource Communication Networks”, *IEEE Transactions on Communications*, Vol. 42, No. 11, Nov. 1994, pp. 2979-2988.
9. C. Chao and W. Chen, “Connection Admission Control for Mobile Multiple-class Personal Communication Networks”, *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 8, 1997, pp. 1618- 1626.
10. I. Yoon and B. Lee, “A Distributed Dynamic Call Admission Control that Supports Mobility of Wireless Multimedia Users”, *Proceedings of the IEEE International Conference on Communications (ICC)*, 1999, pp. 1442- 1446.
11. Y. Xiao, C. Chen, and Y. Wang, “A Near Optimal Call Admission Control with Genetic Algorithm for Multimedia Services in Wireless/Mobile Networks”, *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON)*, Oct. 2000, pp.787-792.
12. N. Bartolini and I. Chlamtac, “Call Admission Control in Wireless Multimedia Networks”, *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (PIMRC), Vol. 1, 2002, pp. 285-289.
13. K. Yeung and S. Nanda, “Channel management in microcell/macrocell cellular radio systems”, *IEEE Transactions on Vehicular Technology*, Vol. 45, Nov. 1996, pp. 601-612.
14. H. Tijms, “Stochastic Modeling and Analysis: A Computational Approach”, Wiley, New York, 1986.
15. K. Hastings, “Introduction to the Mathematics of Operations Research”, Marcel Dekker Inc., 1989.
16. S. Biswas and B. Sengupta, “Call Admissibility for Multirate Traffic in Wireless ATM Networks”, *in Proc. IEEE 1NFOCOM*, Vol. 2, Kobe, Japan, Apr. 1997, pp. 649-657.
17. N. Nasser and H. Hassanein, “Multi-Class Bandwidth Allocation Policy for 3G Wireless Networks", *in Proceedings of the IEEE Conference on Local Computer Networks* (LCN), Bonn, Germany, October 2003, pp. 203-209.
18. G.Y. Zhao, “Interior-Point Methods with Decomposition for Solving Large-Scale Linear Programs”, *Journal of Optimization Theory and Applications*, Vol. 102, No. 1, 1999, pp. 169-192.
19. J.M. Hyman, A.L. Lazar and G. Pacifici, “A separation principle between scheduling and admission control for broadband switching,” *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 4, May 1993, pp. 605-616.

Improving the Performance of Rate Adaptation Schemes in Heterogeneous Wireless Networks

P. Stathopoulos[1], L. Sarakis[1,2], and N. Mitrou[1]

[1] School of Electrical and Computer Engineering,
National Technical University of Athens, Greece
{pstath, ls6}@telecom.ntua.gr, mitrou@softlab.ntua.gr
[2] Institute of Informatics and Telecommunications,
National Center for Scientific Research "Demokritos", Athens, Greece
sarakis@iit.demokritos.gr

Abstract. The emerging heterogeneous networking environment will comprise diverse link layer technologies like 3G, WLANs and WPANs. In this context, multimedia applications should be able to cope with packet losses and delays due to congestion, wireless transmission errors and vertical handovers. This paper addresses the functional requirements from an advanced adaptation architecture, which incorporates enhanced end-to-end adaptation schemes that exploit information delivered by the next-generation terminal. The terminal's management entity should provide information at least for the local access network condition, e.g. estimates of the available bandwidth and wireless link losses, and handover notifications. The adaptation architecture does not count on the existence of QoS infrastructure; however, exploitation of such infrastructure is possible through the collaboration between the terminal and a network management entity. We have extended the LDA and TFRC rate adaptations algorithms to make use of the enhanced functionality. Through simulations we validate the robustness of these extended schemes against vertical handovers.

Keywords: Rate adaptation, Vertical handover, LDA, TFRC.

1 Introduction

In the next-generation networking environment the user will be always connected, selecting from a variety of available, existing and forthcoming network technologies, ranging from 2.5/3G cellular to *Wireless Local Area Networks* (WLANs), *Wireless Personal Area Networks* (WPANs), in home and ad-hoc networks. This environment will be characterized by the heterogeneity and the diversity of the participating networks in terms of performance, *Quality of Service* (QoS) capabilities, and control and management infrastructure. Furthermore, applications are expected to be pervasive; function gracefully and without interruptions, independently of the underlying network technology. In this context, maintaining a high end-to-end service quality, is crucial. The uncertainty

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1297–1308, 2005.

about wide deployment of Internet end-to-end QoS schemes in the near future, suggests that the applications should be able to adapt to changing network conditions.

Adaptation of streaming multimedia applications comes to play when the network state deteriorates, and it is usually triggered by packet losses. Such losses in homogeneous, single technology wireless environments are caused mainly by congestion and wireless transmission errors. In the next-generation environment, however, inter-system mobility and heterogeneity of the involved networks constitute extra sources of packet losses. The several macro and micro mobility protocols that have been proposed ([1], and references therein), result, in practice, to transient packet losses and increased delay, especially during vertical handovers [2]. Even if a seamless handover is offered by the mobility mechanism, severe congestion may arise in the new network, if this has significantly less resources than the old one (upward vertical handover). In such scenarios the appropriate action is to shield the adaptation scheme from transient mobility and resources-heterogeneity-related losses.

Among the proposed TCP-friendly rate adaptation schemes (see [3] for a survey), the *Loss Delay Adaptation* (LDA)[4] algorithm works in conjunction with the RTP/RTCP protocol stack and exploits the RTCP Receiver Reports, in order to transfer feedback information back to sender, while the *TCP-Friendly Rate Control* algorithm (TFRC)[5] takes into consideration feedback messages sent every *Round Trip Time* (RTT). These schemes require several RTTs and several seconds, respectively, to adjust their sending rate. In case of vertical handovers, this delay could cause buffers to temporarily saturate in the new network. To the best of our knowledge, the effects of vertical handovers on the performance of rate adaptation schemes have not been studied in sufficient depth.

The goal of this paper is twofold: first, to identify the functional entities that are expected to have a key role in an adaptation architecture targeting at next-generation heterogeneous networks, and second, to exploit this enhanced functionality towards making rate adaptation schemes robust against wireless corruption errors and vertical handovers. For evaluation proposes we have chosen to extend the LDA and the TFRC, mainly because they are representative examples of application and transport layer adaptation schemes, respectively.

The rest of paper is organized as follows. Section 2 briefly reviews existing work on rate adaptation. Section 3 identifies the functional components of the adaptation architecture. The extensions to the LDA and the TFRC algorithms are described in Section 4, and are validated against simulation in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

For applications running over wireless links the ability to discriminate between losses caused by congestion and losses caused by poor link quality is of great value. For losses caused by poor link quality the most appropriate action is to rely on error control (through link layer retransmissions and forward error control schemes) and adaptation mechanisms, like bandwidth compression using scalable coding or adaptive packetization. Other approaches propose to use information from several layers to derive the fraction of lost packets in the wireless link. One such mechanism, ([6]), introduces a monitoring agent in the boundary of the wired/wireless segments in order to discriminate between packet drops due to congestion in the wired segment, and losses that are due to wireless transmission errors. Along the same lines, the authors in [7] use a TCP throughput formula to adjust the sending rate. [8] presents two TCP-friendly adaptive transport layer protocols: one for reliable data transmission and one for UDP-based multimedia transmission. Both are based on an "additive increase multiplicative decrease" algorithm with variable parameters. Each protocol represents a single unified approach that can be used equally well for heterogeneous networks with diverse characteristics in terms of delay and packet loss. The results comment on the mean throughput and the fairness achieved in the steady state and do not provide details on the performance of the schemes soon after the handover.

To the best of our knowledge, [9] is the only work that evaluates the performance of TFRC in the presence of vertical handovers (however, it elaborates only on the throughput achieved after the handover). This work, which has apparently been developed independently of ours (the motivation, including an initial architecture and some preliminary results, for our work lies in [10]) shows that the performance of TFRC can severely degrade during handovers and propose two mechanisms to overcome the problem (overbuffering and explicit handover notification).

3 Abstraction of the Adaptation Architecture

The majority of the adaptations schemes, briefly reviewed in the previous section, assume that certain entities will exist in the next-generation heterogeneous environment and reside both at the terminal and network sides. As far as the former is concerned, the requirements include the presence of a *Terminal Management Entity* (TME) and an appropriate middleware through which enhanced applications will make use of the functionality provided by the TME. The core functionality of TME consists in performing network availability detection, gathering link statistics and performing network condition estimation for vertical handover decision. As far as the network side is concerned, the presence of a *Network Management Entity* (NME) is highly encouraged. In this case, the collaboration between the NME and the terminal can lead to optimal handover decisions.

The proposed architecture combines end-to-end adaptation schemes with feedback messages that are enriched so as to include information provided by both the sender's and receiver's TMEs. The information generated by the TMEs is required to at least correspond to the status of the access networks or, complementary, to a more global network status, should the TMEs cooperate with corresponding NMEs. This enriched feedback information is furnished to the adaptation scheme which, in turn, calculates appropriate adaptation targets in terms of target rate and protection level against wireless errors. Based on these parameters, the streaming application adapts appropriately by adjusting the encoding properties of the media stream.

At a glance, the sophisticated adaptation architecture incorporates: a) the employment of efficient end-to-end adaptation schemes that do not require any QoS scheme to exist, b) the exploitation of the TME access network monitoring information, both at the sender's and receiver's sides, c) the definition of enhanced feedback messages and d) the exploitation of established QoS mechanisms through the NME and TME interaction. The high level concept is to combine the strengths of the end-to-end schemes, namely, robustness, scalability, network transparency and simplicity, with the minimum set of functionalities that the next-generation networking elements will offer. QoS facilities should not be taken for granted for the entire path; this assures that the adaptation architecture is viable even when only best-effort service is provided.

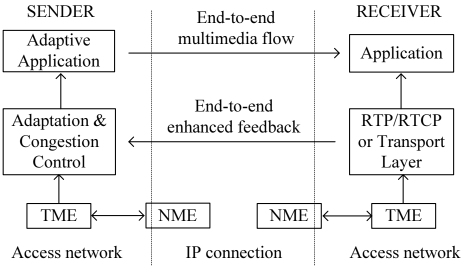

SENDER
End-to-end multimedia flow
RECEIVER
Adaptive Application
Application
End-to-end enhanced feedback
Adaptation & Congestion Control
RTP/RTCP or Transport Layer
TME
NME
NME
TME
Access network
IP connection
Access network

Fig. 1. Abstraction of the adaptation architecture

The adaptation architecture is depicted in Fig. 1. The end-to-end feedback messages alone can not provide discrimination between drops due to congestion and losses due to the quality of the wireless link. This desired functionality is provided by the TME. The information from the receiver's TME, combined with end-to-end information, like packet loss, RTT and delay variation of the complete path between the sender and the receiver, is sent back to the sender as an enhanced feedback message. This message together with the information provided by the local TME feed the congestion control and adaptation module. The TME can offer more than just local access network information when it cooperates with an NME.

In greater detail, the TME, by monitoring the present networks, can estimate, even in a coarse manner, the available resources. Furthermore, since the TME participates in network selection and handover decisions, it already has knowledge about imminent vertical handovers. Thus, the TME must calculate/retrieve and report to the adaptation module the following parameters:

- The R_{AVL} and, optionally, the R_{MAX}. R_{AVL} is an estimation of the currently available rate at the IP layer in the current network conditions (in case of handover the estimation can be proactive). R_{MAX} is the nominal value of the maximum rate at the IP layer the current network environment can support. The granularity in estimating R_{AVL} and R_{MAX} can vary and depends on the capabilities of the TME.

- The l_W. This value denotes the fraction of packets lost due to wireless errors during the last reporting period.
- The boolean flag h. This indicates whether a vertical handover is underway, and should be reported as soon as a vertical handover is decided by the TME.

  Both sender's and receiver's TMEs must provide the adaptation module with the information vector $TME_{out} = (R_{AVL}, R_{MAX}, l_w, h)$. The adaptation schemes employed should compensate for inconsistencies in the reported TME_{out} values. The following characteristics determine both the TME and the adaptation scheme requirements. In this respect:
- There is no requirement from the TME *regarding the precision* of the provided R_{AVL} value. This value must be used only as an indication, and the adaptation schemes must be able to recover gracefully from inaccurate estimations.
- The R_{AVL} and R_{MAX} values can correspond to end-to-end or local access network values.
- Updated R_{AVL} and R_{MAX} values are required to be reported in vertical handover events. These values are not required to be periodically reported, and thus reflect dynamic changes inside homogeneous networks, since the end-to-end adaptation schemes should be able to handle such changes.
- End-to-end QoS provisioning must not be required or taken for granted by the adaptation scheme.

Thus, the adaptation schemes used within the context of this architecture should be able to cooperate with TMEs that report TME_{out} information based on the aforementioned requirements.

4 The Extended Adaptation Schemes

Two schemes are proposed in order to fit in the described architecture. The first is based on the LDA and the second on the TFRC (referred to as ext-LDA and ext-TFRC, respectively). These schemes are able to resolve losses due to wireless corruption errors and losses during vertical handovers, whether the latter are due to the mobility mechanism or the heterogeneity of the network's resources.

4.1 The Extended LDA

The analysis follows that in [4]. (Due to lack of space we focus only on the differences between LDA and ext-LDA.) In LDA, the sender target rate is estimated by taking into account the fraction of lost packets, l, the round trip time, and the bottleneck bandwidth. The ext-LDA using local TME information (i.e. the vector TME_{out}) isolates losses due to wireless errors. If the fraction of lost packets at the sender's wireless network, l_W^{snd}, and at the receiver's access network, l_W^{rcv}, are known, the fraction of congestion losses, l_c, is given by the formula: $l_c = l - (l_W^{snd} + l_W^{rcv})$.

If $l_c > 0$, the sender reacts by decreasing the target rate, similar to the original LDA scheme (for simplicity we assume unicast transmissions). If r is the current rate, the new rate r' is given by $r' = r(1 - l_c \cdot R_f)$, where R_f is a reduction factor determining the degree of the sender's reaction to losses. During periods of no congestion or vertical handovers, ext-LDA increases the target rate in the same way that LDA does (i.e. by using the Additive Increase Rate (AIR) parameter). When a vertical handover occurs, the node performing this handover sends, without waiting for the reporting timer to expire, its TME vector to the adaptation module. (The sender uses the interface between the adaptation module and the TME, while the receiver uses feedback messages.) The adaptation module inspects the boolean flag h and, in case of a reported handover, calculates a new target rate r_{new}:

$$r_{new} = \min(R_{AVL}, r_{ext-LDA}),$$

where R_{AVL} is the estimation of the available bandwidth of the new network and $r_{ext-LDA}$ is the ext-LDA rate that would otherwise be calculated at that instance, if vertical handover did not occur. In this way, when the terminal moves to a network with fewer available resources ext-LDA reacts with an immediate rate change. If, on the other hand, the new network has more available bandwidth, the sending rate is increased smoothly according to the ext-LDA algorithm. In any case, after a handover the AIR parameter of the algorithm is set to its initial value.

In the ext-LDA algorithm the sender ignores the first regular RTCP feedback report sent by the receiver after the handover. This is because packet losses reported immediately after a handover cannot provide clear congestion indication. Using this heuristic mechanism, losses that are due to the mobility mechanism are isolated. The value l_W, carried in the information vectors TME_{out} can be used to feed a fidelity module that calculates the protection level against transmission errors.

4.2 The Extended TFRC

The original TFRC algorithm ([5]) uses a TCP throughput equation to calculate the streaming rate. The loss event rate is calculated at the receiver and made known to the sender through feedback messages issued once every RTT. Using these feedback messages the sender estimates the value of the RTT and, implicitly, the value of retransmission timeout. The TFRC incorporates a sophisticated algorithm to map packet losses to loss events in order to obtain the loss event rate, p. If $p > 0$ the updated sending rate is the minimum between the rate that has been calculated by the TCP throughput equation and twice the receiver's rate. When there are no losses the sending rate is approximately doubled every RTT.

The ext-TFRC, like the ext-LDA, takes also into account the information vectors TME_{out} that are provided by the sender's and receiver's TMEs. The receiver's TME information is incorporated into augmented TFRC feedback messages. If we relax the condition that the TFRC algorithm reacts according to the rate of loss events and we allow it to react according to the fraction of lost packets then, by using the l_W information in TME_{out}, it is possible to make it react only to congestion losses (in a fashion similar to ext-LDA).

When the receiver performs a vertical handover it issues an urgent TFRC feedback message notifying the sender about the parameters of the new network through the TME_{out}^{rcv} vector. If case of sender's handover it is assumed that its TME has already knowledge of the parameters of the new network. In any case the new sending rate is given by:

$$r_{new} = \min(R_{AVL}, r_{ext-TFRC}).$$

After the handover, the sender clears the history related to RTT and loss event rate estimates since this does not represent the current network conditions. We have chosen to let ext-TFRC keep the rate r_{new} until the first valid feedback message is received (we have heuristically chosen to discard the first two feedback messages after the handover; therefore, the first valid message is the third one). This is done in order to filter out excessive losses and large RTTs due to the mobility mechanisms.

5 Simulation Results

In this section we evaluate the performance of the extended schemes against the original ones and TCP using the *ns-2* simulator. The ext-TFRC is also compared to a variation of TFRC that resets its congestion state (starting eventually from slow start) after vertical handovers. This enhancement has been proposed in [11] and will be referred to as ext-TFRC/SS.

5.1 Simulation Topology and Performance Metrics

The topology that is used is depicted in Fig. 2. There are two access networks, namely Net-A and Net-B, each of which has four hosts ($H = F = 4$). Initially, a mobile host (MH) is served through Net-A. After a warm-up period, MH performs a handover from Net-A to Net-B (time instance t_1). Inter-system handover is supported by the Mobile IP v4 protocol ([12]). The common resources on both networks are shared using a generic round robin scheduler to achieve fairness in bandwidth sharing. The packet size is set to 1kbyte. All paths, except GW-HA and GW-FA have transfer delays equal to $1ms$. The links GW-HA and GW-FA

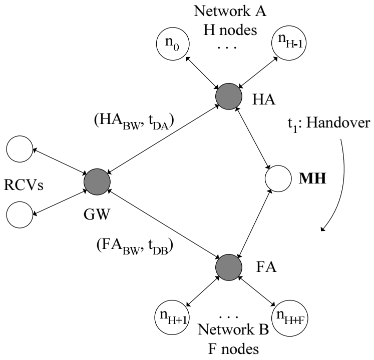


Fig. 2. The simulation topology. HA: Home agent, FA: Foreign agent

represent the bottleneck links; all other links are over-provisioned. All queues, except HA-GW and FA-GW are $50Kbytes$ long. It is assumed that the MH is able to estimate, prior to handover, the bandwidth that it will get in the new network. This estimate is provided by the TME.

The precision with which the TME estimates the available bandwidth is denoted by the precision factor, f, as the ratio of the true to the estimated bandwidth that is available to MH (i.e. $f = 0.5$ denotes bandwidth overestimation by a factor of two). It is further assumed that the access networks are reliable links and thus no transmission errors occur.

For every source, the actual sending rate, rs_i, the throughput, r_i, and the fraction of lost packets, d_i, are calculated. The first performance metric is the ratio of the source's sending rate to its fair share of the Net-B bandwidth, given by $\hat{rs_i} = rs_i/r_{fair}$, $r_{fair} \overset{def}{=} FA_{BW}/(F+1)$. The second metric is the total utilization $U(r,d)$ of a system with N sources and is given by $U(r,d) = (\sum_i^N U_i(r_i)g_i(1-d_i))/N$, where $U_i(r_i)$ is the utilization function of source i and $g_i(1-d_i)$ is a non linear factor that accounts for utilization reduction due to packet losses d_i ([13], [14]). A utilization function that has the desirable properties is given in [14] by $U_i(r) = log_2(r_i)$. Equivalently, in this paper we use the utilization function $U_i(r) = log_2(\hat{r_i}+1)$, which delivers a utilization equal to one when the distribution of the throughput is fair ($\hat{r_i}$ is the ratio of throughput to its fair share). Similarly to [13], the non linear function $g_i(z)$ is approximated by $g_i(z) = z^2$, $\forall i$.

The third metric is the fairness index of the bandwidth distribution in Net-B after the handover. This index is given by ([15]) $fi = (\sum_{i=1}^N x_i)^2/(N\sum_{i=1}^N x_i^2)$, $x_i = \hat{r}_i$. The last metric of interest is the Coefficient of Variation, CoV, of the sending rate of MH, rs_{MH}. If the average value of the sending rate calculated over a t_w time window after the handover is $\overline{rs}_{MH}^{t_w}$, and the standard deviation is $\sigma(rs_{MH}^{t_w})$, the CoV is given by $CoV(rs_{MH}^{t_w}) = \sigma(rs_{MH}^{t_w})/\overline{rs}_{MH}^{t_w}$.

5.2 Performance Evaluation of ext-LDA

In the first set of simulations $FA_{BW} = 256kbps$, $t_{d_A} = t_{d_B} = 512ms$, $t_1 = 900s$, the queue length HA-GW corresponds to $512ms$ delay, and the queue length FA-GW is $16Kbytes$. Figure 3 shows that the LDA reacts slowly to the handover event and keeps its high sending rate for a considerable amount of time. The rate is then dropped abruptly as a response to reported packet drops. The result is that for an observation window of $60s$ after the handover the LDA makes little use of its available bandwidth. The ext-LDA with $f = 1$ accurately adapts to its fair share (as it should by assumption), with $f = 0.5$ it drops its sending rate at a fairly slow pace, achieving finally a fair share, and for $f = 2$ it delivers a smoother rate increase than LDA. Fig. 4 shows the CoV (covariance) of the sending rate for $t_w = 60s$. The LDA has the larger CoV for any HA_{BW}. The mean values of fairness and utilization indices, calculated for $t_w = 30s$, are depicted in Fig. 5 (the figure includes the 95% confidence intervals for the extended schemes). The graph shows that the ext-LDA outperforms LDA, especially when the bandwidth heterogeneity between the two networks is large.

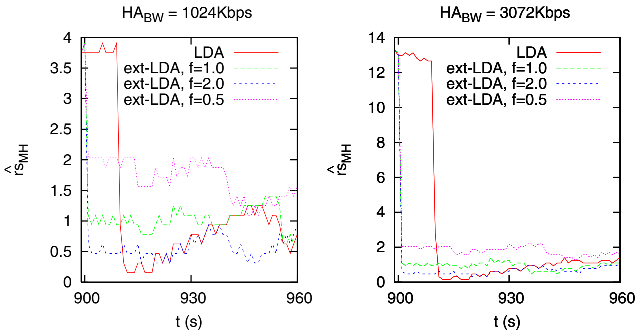


Fig. 3. Normalized sending rate $\hat{rs}_{MH}$ vs. time (LDA competing flows in Net-B)

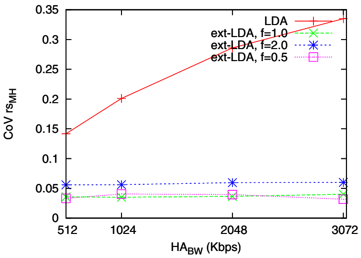


Fig. 4. Covariance of MH's sending rate vs. HA_{BW} (LDA competing flows in Net-B)

We have also compared the performance of ext-LDA to that of LDA in the presence of competing TCP flows in Net-B (the figures are not included due to lack of space). The results were similar and showed that the ext-LDA schemes preserve their small rate variation and achieve far better utilization and fairness. However, due to the relatively larger rate variation of the TCP flows, the sending rate of ext-LDA delivers slightly larger figures for the CoV.

5.3 Performance Evaluation of ext-TFRC

In the second set of simulations $FA_{BW} = 512kbps$, $t_{d_B} = 250ms$, $t_1 = 200s$, the queue length HA-GW corresponds to $384ms$ delay and the queue length FA-GW is $25Kbytes$. The reason for using a larger FA_{BW} is because TFRC makes better use of large available bandwidths. Variable t_{d_A} values are used to simulate networks with diverse bandwidth-delay products and provide insights into the impact the retransmission mechanisms have on the performance of TFRC. Figure 6 shows the evolution of the TFRC, ext-TFRC and ext-TFRC/SS sending rates after the MH's handover to Net-B, where four TFRC sessions are in progress. All schemes are prevented from converging to their fair share of the bandwidth at a satisfactorily fast pace. This is due to the large value of RTT at Net-A. Simulations with $t_{d_A} = 5, 250ms$ delivered a significantly smaller convergence period (small RTTs give rise to TFRC timers expiration and, consequently, to rate reduction,

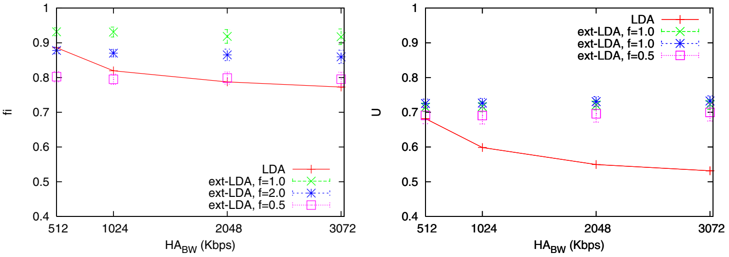


Fig. 5. Mean value of fairness index (left) and utilization (right) vs. HA_{BW} (LDA competing flows in Net-B)

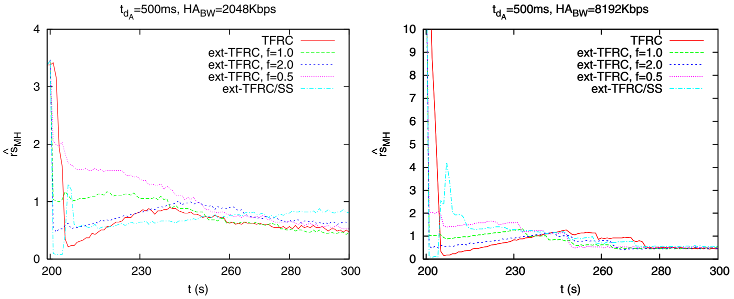


Fig. 6. Normalized sending rate $\hat{rs}_{MH}$ vs. time for $t_{d_A} = 500ms$ (TFRC competing flows in Net-B)

which is, in the case of upward handover, beneficial). Figure 6 illustrates that the TFRC reacts more slowly to network changes compared to ext-TFRC and ext-TFRC/SS. Shortly after the handover, the ext-TFRC/SS suffers from unstable behavior demonstrated by abrupt rate changes. The ext-TFRC with $f = 1$ and $f = 2$ seems to obtain the best performance demonstrated through small rate variations. The ext-TFRC with $f = 0.5$ cannot sustain a high sending rate but still avoids abrupt rate changes. Figure 7 shows the CoV of the sending rate for $t_w = 60s$ and $t_{d_A} = 5,500ms$. In most of the settings, the ext-TFRC schemes outperform the original TFRC and ext-TFRC/SS. These observations still hold true for the mean value of the fairness and utilization indices, depicted in Fig. 8 ($t_w = 30s$). The performance of ext-TFRC has been also studied against TCP flows. The simulations, not depicted due to space limitations, illustrated that both the TFRC and the ext-TFRC were more aggressive and managed to obtain more than their fair share of bandwidth (for several seconds after the handover), especially for large HA_{BW} and small t_{d_A}. Nevertheless, the ext-TFRC schemes with $f = 1$ and $f = 2$ still performed better than the TFRC and the ext-TFRC/SS.

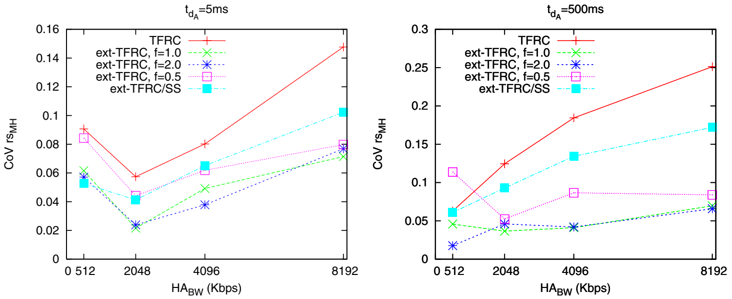


Fig. 7. Covariance of MH's sending rate vs. HA_{BW} for $t_{d_A} = 5, 500ms$ (TFRC competing flows in Net-B)

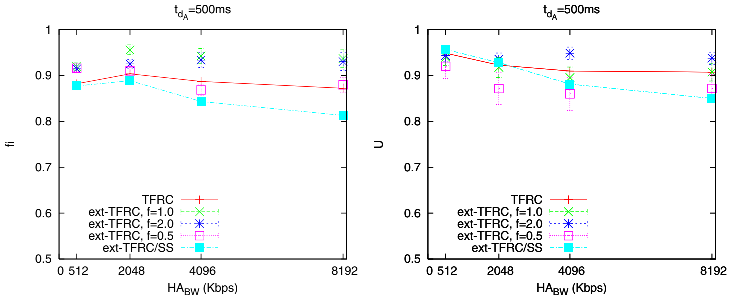


Fig. 8. Mean value of fairness index (left) and utilization (right) for $t_{d_A} = 500ms$ (TFRC competing flows in Net-B)

6 Conclusions

In this paper we presented an adaptation architecture targeting at next-generation heterogeneous networks and extended the LDA and TFRC rate adaptation schemes to fit in this architecture. The most essential feature of the architecture is the cooperation between a management entity, located at the terminal, and end-to-end adaptation schemes. This management entity provides enhanced feedback to the extended adaptation schemes, including estimates of available bandwidth and losses due to poor link performance, and handover notifications. This approach targets at a low implementation overhead and high deployment scalability; most tasks are handled by end point nodes, and no special entities, like RTP monitoring agents, are required in the wired/wireless networks boundaries. The simulation results showed that the extended versions delivered better performance in terms of responsiveness, rate stability, utilization and fairness. The improved performance was maintained even when terminal management entities did not report perfectly accurate available-bandwidth figures, provided that these were not heavily overestimated. Future work will address a more detailed evaluation of the implementation overhead of the proposed schemes and study their performance in simulation scenarios featuring more complex topologies with a greater number of hosts.

References

1. Reinbold, P., Bonaventure, O.: A Comparison of IP Mobility Protocols, Tech. Rep. Infonet-TR-2001-07, University of Namur, Infonet Group (2001)
2. Stemm, M., Katz, R.H.: Vertical handoffs in wireless overlay networks. Mobile Networks and Applications **3** (1998) 335–350
3. Widmer, J., Denda, R., Mauve, M.: A survey on TCP-friendly congestion control. IEEE Network **15** (2001) 28–37
4. Sisalem, D., Schulzrinne, H.: The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In: Proc. of NOSSDAV, Cambridge, UK. (1998)
5. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-based congestion control for unicast applications. In: Proc. of SIGCOMM, Stockholm, Sweden (2000) 43–56
6. Yoshimura, T., Kawahara, T., Ohya, T., Etoh, M.: QoS Control Architecture with RTP Monitoring Agent for Mobile Multimedia Streaming. In: IPSJ Symposium on Multimedia, Distributed, Cooperative and Mobile Systems. (2001)
7. Yang, F., Zhang, Q., Zhu, W., Zhang, Y.Q.: End-to-end tcp-friendly streaming protocol and bit allocation for scalable video over wireless internet. IEEE JSAC **22** (2004) 777–790
8. Akan, O.B., Akyildiz, I.F.: ATL: An adaptive transport layer suite for next-generation wireless internet. IEEE JSAC **22** (2004) 802–816
9. Gurtov, A., Korhonen, J.: Effect of vertical handovers on performance of tcp-friendly rate control. ACM Mobile Computing and Communications Review **8** (2004) 73–87
10. Stathopoulos, P., Papageorgiou, P., Kouis, D., Zoi, S., Mitrou, N.: A rate adaptation scheme for media streaming over heterogeneous networks. International Journal of Wireless and Mobile Computing (2005) to appear in issue 5.
11. Kohler, E., Handley, M., Floyd, S.: Datagram Congestion Control Protocol (DCCP), Internet Draft, IETF (2004)
12. Perkins, C.: Mobility support for IPv4, RFC 3220, IETF (2002)
13. Bajaj, S., Breslau, L., Shenker, S.: Uniform versus Priority Dropping for Layered Video. In: Proc. of SIGCOMM. (1998) 131–143
14. Mo, J., Walrand, J.: Fair end-to-end window-based congestion control. IEEE/ACM Transactions on Networking **8** (2000) 556–567
15. Jain, R.: The Art of Computer Systems Performance Analysis. Wiley (1991)

Stochastic Admission Control for Quality of Service in Wireless Packet Networks

Majid Ghaderi[1], Raouf Boutaba[1], and Gary W. Kenward[2]

[1] University of Waterloo, Waterloo, ON N2L 3G1, Canada
{mghaderi, rboutaba}@uwaterloo.ca
[2] Nortel Networks, Ottawa, ON K1Y 4H7, Canada
gkenward@nortelnetworks.com

Abstract. Call admission control is a key element for providing quality of service (QoS) in mobile wireless networks. Traditional admission control schemes only address call-level QoS guarantee because of the underlying circuit-based network architecture. In contrast, emerging wireless technologies such as 3G and 4G tend to be packet-switched rather than circuit-switched because the packet-based architecture allows better sharing of the scarce wireless resources. This paper introduces a novel distributed call admission control scheme called PFG, which maximizes the wireless channel utilization subject to a predetermined bound on the call dropping and packet loss probabilities for variable-bit-rate traffic in a packet-switched wireless cellular network. In particular, we show that in wireless packet networks, the undesired event of dropping an ongoing call can be completely eliminated without sacrificing the bandwidth utilization. Extensive simulation results show that our scheme satisfies the hard constraint on call dropping and packet loss probabilities while maintaining a high bandwidth utilization.

1 Introduction

Emerging wireless technologies such as 3G and 4G [1, 2] tend to be packet-switched rather than circuit-switched because the packet-based architecture allows better sharing of limited wireless resources. In a packet network, calls do not require dedicated circuits for the entire duration of connection. Unfortunately, this enhanced flexibility makes it more difficult to effectively control the admission of connections into the network.

In wireless packet networks there exist two levels of quality of service, namely, call-level and packet-level. At call-level, two important parameters which determine the quality of service are *call blocking probability* and *call dropping probability*. Since dropping a call in progress has more negative impact from the user perspective, handoff calls are given higher priority than new calls in access to wireless resources. At packet-level, *packet loss probability*, delay and jitter are the most important QoS parameters. There is always a trade-off between the network utilization and the QoS perceived by users. It is desired to have a re-

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1309–1320, 2005.

source allocation scheme which can satisfy the prespecified QoS constraints while maximizing the utilization of the network resources.

Call admission control (CAC) has been extensively studied in circuit-switched (voice) wireless cellular networks (see [3, 4, 5] and references there in). Hong and Rappaport [6] are the first who systematically analyzed the famous *guard channel* (GC) scheme, which is currently deployed in cellular networks supporting voice calls. Ramjee et al. [7] showed that the guard channel scheme is optimal for minimizing a linear objective function of call blocking and dropping probabilities while the *fractional guard channel* scheme (FG) is optimal for minimizing call blocking probability subject to a hard constraint on call dropping probability. Instead of explicit bandwidth reservation as in the GC, the FG accepts new calls according to a randomization parameter called the *acceptance ratio*. Because of user mobility, it is impossible to describe the state of the system by using only local information, unless we assume that the network is uniform and approximate the overall state of the system by the state of a single cell in isolation. To include the global effect of mobility, collaborative or distributed admission control schemes have been proposed [8, 9, 10]. Information exchange among a cluster of neighboring cells is the approach adopted by all distributed schemes.

None of these papers has considered a wireless packet-switched network. There is no packet-level consideration in these works. Call dropping and blocking probabilities are the only QoS parameters considered. In circuit-switched networks, when a handoff call arrives while there is no idle circuit (wireless channel), the handoff fails and hence the call is dropped. In contrast, in a packet-switched network it is still possible to accept the handoff call at the expense of probably increasing the number of dropped packets. While this approach completely eliminates the call dropping event, we will show that its impact on packet loss can be effectively controlled.

We introduce a *packetized fractional guard channel* (PFG) call admission control mechanism for cellular packet networks that achieves a high bandwidth utilization while satisfying a target packet loss probability without dropping any ongoing call. To the best of our knowledge, PFG is the first to address the combination of call-level and packet-level QoS while *explicitly considering the mobility of users*. The main features of PFG are as follows:

1. PFG achieves zero percent call dropping,
2. PFG is dynamic, therefore, adapts to a wide range of system parameters and traffic conditions,
3. PFG is distributed and takes into consideration the information from direct neighboring cells in making admission decisions,
4. The control mechanism is stochastic and periodic to reduce the overhead associated with distributed control schemes.

The motivation behind this study is to support variable-bit-rate (VBR) multimedia traffic in emerging wireless packet cellular networks. The rest of the paper is organized as follows. Our system model, assumptions and notations are described in section 2. Section 3 describes the high-level operation of the proposed admission control algorithm while in section 4, detailed analysis of the algorithm is presented. Simulation results are presented in section 5 and section 6 concludes this paper.

2 System Model

The considered system is a packet-switched cellular network, in which the users move along an arbitrary topology of $\mathcal{B}$ cells according to the routing probabilities r_{ij} (from cell i to cell j). Each cell i has a set of adjacent cells denoted by $\mathcal{A}_i$. We assume that there is one type of calls in the system. Although the dynamic behavior of wireless channel may cause packets to be dropped, we assume that there are appropriate underlying coding and retransmission mechanisms to cope with packet loss due to channel effects. Therefore, cell overflow, receiving more traffic than what can be actually transmitted, is considered the only source of packet loss. This paper considers constant cell capacity, however, the approach that we propose next can be extended to include variable capacity cases using a technique similar to the one proposed in [11]. Moreover, we assume that

1. New call arrivals to a cell are independent and Poisson distributed with rate λ_i.
2. Cell residency times are independent and exponentially distributed with mean $1/h$. However, we show that the proposed algorithm is insensitive to this assumption.
3. Call durations are independent and exponentially distributed with mean $1/\mu$.

These assumptions are widely used in literature [4,5,3,6,7,8,9,10] for the mean value analysis of cellular systems.

2.1 Maximum Occupancy in a Cell

Let M_i denote the maximum occupancy, i.e. maximum number of calls, in cell i under the so-called *average bandwidth assignment* scheme. This scheme allocates to each VBR call a share of bandwidth equal to the call's average bandwidth requirement. Let m denote the average bandwidth requirement of a call, then

$$M_i = \frac{c_i}{m}, \tag{1}$$

where c_i denoted the capacity of cell i. Although this scheme achieves a high bandwidth utilization, it leads to a high rate of packet loss [12]. If there are more than M_i calls in cell i, then we say that the cell is in *overloaded state.* In the overloaded state, probability of packet loss is very high. Our scheme, PFG, rejects new call requests when a cell is in overloaded state.

2.2 Time-Dependent Handoff Probability

Let $P_h(t)$ denote the probability that a call hands off to another cell by time t and remains active until t, given that it has been active at time 0. Also, let $P_s(t)$ denote the probability that a call remains active in its home cell until time t, given that it has been active at time 0. It is obtained in [13] that $P_h(t) = (1 - e^{-ht})\, e^{-\mu t}$ and $P_s(t) = e^{-(\mu+h)t}$. On average, for any call which arrives at time $t' \in (0, t]$, the average handoff and stay probabilities $\tilde{P}_h$ and $\tilde{P}_s$ are expressed as

$$\tilde{P}_h(t) = \frac{1}{t} \int_0^t P_h(t - t')\, dt' , \tag{2}$$

$$\tilde{P}_s(t) = \frac{1}{t} \int_0^t P_s(t - t')\, dt' . \tag{3}$$

Similar to [8] and [9], we assume that during a control period each call experiences at most one handoff. This assumption is justified by setting the length of the control period T reasonably shorter than the average cell residency time. Discussion on the appropriate control interval is not included here due to paper length restriction. For the optimal control interval T, please refer to [13].

Finally, let $P_{ji}(t)$ denote the time-dependent handoff probability that an active call in cell j at time 0 will be in cell i at time t, where $j \in \mathcal{A}_i$. Since each call experiences at most one handoff during the control period, it is obtained that $P_{ji}(t) = P_h(t)\, r_{ji}$. Similarly, the average handoff probability $\tilde{P}_{ji}(t)$ for a call which arrives at any time $t' \in (0, t]$ is given by $\tilde{P}_{ji}(t) = \tilde{P}_h(t)\, r_{ji}$.

3 Admission Control Algorithm

The proposed distributed algorithm, PFG, consists of two components. The first component is responsible for retrieving the required information from the neighboring cells and computing the acceptance ratio. Using the computed acceptance ratio, the second component enforces the admission control locally in each cell. The following sections describe these two components in detail.

3.1 Distributed Control Algorithm

To reduce the signalling overhead all the information exchange and acceptance ratio computations happen only once at the beginning of each control period of length T. Several steps involved in PFG distributed control are described below:

1. At the beginning of a control period, each cell i sends the number of active calls in the cell at the beginning of the control period denoted by $N_i(0)$ to its adjacents and receives the number of new calls, N_i, which were admitted in the last control period by each adjacent cell.
2. Now, cell i uses the received information and those available locally to compute the acceptance ratio a_i using the technique described in section 4.
3. Finally, the computed acceptance ratio a_i is used to admit call requests into cell i using the algorithm presented in section 3.2.

3.2 Local Control Algorithm

Let s_i denote the state of cell i, where there are s calls active in the cell. Let $a_i(s)$ denote the acceptance ratio where the cell state is s_i. Fig. 1 shows the state transition diagram of the PFG scheme in cell i. In this diagram, ν_i is the handoff arrival rate into cell i, and M_i is the maximum occupancy given by (1).

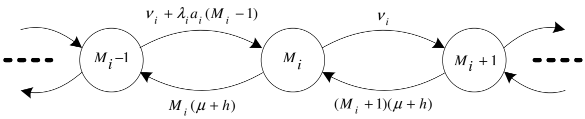


Fig. 1. Packetized fractional guard channel transition diagram

The pseudo-code for the local admission control in cell i is given by the algorithm in Fig. 2. In this algorithm, x is a call requesting a connection into cell i. The acceptance ratio for the respective control period is a_i. Also, `rand(0, 1)` is the standard uniform random generator function. In the next section, we will present a technique to compute the acceptance ratio a_i in order to complete this algorithm.

```
if ($x$ is a handoff call) then
  accept call;
else /* $x$ is a new call */
  if (rand(0,1) $< a_i$) & ($N_i(t) \leq M_i$) then
    accept call;
  else
    reject call;
  end if
end if
```

Fig. 2. Local call admission control algorithm in cell i

4 Computing the Acceptance Ratio

Assuming the target loss probability is sufficiently small, we approximate the packet loss probability by the overflow probability in each cell. This is often an overestimate of the actual buffer overflow probability since it ignores the smoothing effect of the buffer, i.e. the buffer allows the arrival rate to exceed the service rate for short periods. The significance of such inaccuracies must be tempered by the fact that even an exact model does not provide a correct measure of the loss probability seen by calls, as it can not fully capture the impact of interactions within the network. This is a common technique in approximating packet loss probability (see for example [12]). However, as the overflow probability decays to zero, both measures converge to the same value and the difference becomes negligible. Therefore, the time-dependent packet loss probability at time t in cell i denoted by $L_i(t)$ is given by

$$L_i(t) = \Pr\big(R_i(t) > c_i\big), \tag{4}$$

where $R_i(t)$ denotes the total (new and handoff) packet arrival rate into cell i at time t. The main idea is to describe $R_i(t)$ using a Gaussian distribution. The motivation behind Gaussian traffic characterization is that it is very natural when a large number of traffic sources are multiplexed (motivated by the central limit theorem), as is expected to be the case in future wireless networks.

4.1 Traffic Characterization

Let r_n denote the packet generating process of an individual call n. It is assumed that individual packet generating processes are independent and identically distributed (iid) random variables with the mean and variance $E[r]$ and $V[r]$, respectively. Then, $R_i(t)$, the total packet arrival rate in cell i at time t, is expressed as

$$R_i(t) = \sum_{n=1}^{N_i(t)} r_n, \tag{5}$$

where $N_i(t)$ denotes the number of calls at time t. To characterize $R_i(t)$ by a Gaussian distribution we need to specify the parameters of $R_i(t)$, namely, mean and variance. Using the moment generating functions of random processes $R_i(t)$ and r_n, it is obtained that (please refer to [13] for details)

$$E[R_i(t)] = E[N_i(t)]E[r], \tag{6}$$

$$V[R_i(t)] = E[N_i(t)]V[r] + V[N_i(t)]E^2[r]\,. \tag{7}$$

Given that $E[r]$ and $V[r]$ are first order statistics, they can be estimated from measured traffic data which makes this traffic characterization ideal from a measurement point of view [14]. Also, individual packet generating processes can have arbitrary correlation structure and this includes self-similar processes as well.

4.2 Mobility Characterization

The number of calls in cell i at time t is affected by two factors: (1) the number of background (existing) calls which are already in cell i or its adjacent cells, and, (2) the number of new calls which will arrive in cell i and its adjacent cells during the period $(0, t]$ $(0 < t \leq T)$. Let $g_i(t)$ and $n_i(t)$ denote the number of background and new calls in cell i at time t, respectively.

A background call in cell i will remain in cell i with probability $P_s(t)$ or will handoff to an adjacent cell j with probability $P_{ij}(t)$. A new call which is admitted in cell i at time $t' \in (0, t]$ will stay in cell i with probability $\tilde{P}_s(t)$ or will handoff to an adjacent cell j with probability $\tilde{P}_{ij}(t)$. Therefore, the number of background calls which remain in cell i and the number of handoff calls which come into cell i during the interval $(0, t]$ are binomially distributed. Using this property, the time-dependent variance of stay and handoff processes denoted by $V_s(t)$ and $V_{ji}(t)$ and their average counterparts denoted by $\tilde{V}_s(t)$ and $\tilde{V}_{ji}(t)$ can be computed (please refer to [13] for details).

The number of calls in cell i is the summation of the number of background calls, $g_i(t)$, and new calls, $n_i(t)$. Therefore, the mean number of active calls in cell i at time t is given by

$$E[N_i(t)] = E[g_i(t)] + E[n_i(t)], \tag{8}$$

where,

$$E[g_i(t)] = N_i(0)P_s(t) + \textstyle\sum_{j\in\mathcal{A}_i} N_j(0)P_{ji}(t), \tag{9}$$

$$E[n_i(t)] = (a_i\lambda_i t)\tilde{P}_s(t) + \textstyle\sum_{j\in\mathcal{A}_i}(a_j\lambda_j t)\tilde{P}_{ji}(t)\,. \tag{10}$$

Similarly the variance is given by

$$V[N_i(t)] = V[g_i(t)] + V[n_i(t)], \tag{11}$$

where,

$$V[g_i(t)] = N_i(0)V_s(t) + \textstyle\sum_{j\in\mathcal{A}_i} N_j(0)V_{ji}(t), \tag{12}$$

$$V[n_i(t)] = (a_i\lambda_i t)\tilde{V}_s(t) + \textstyle\sum_{j\in\mathcal{A}_i}(a_j\lambda_j t)\tilde{V}_{ji}(t)\,. \tag{13}$$

Note that given the arrival rate λ_i and the acceptance ratio a_i, the *actual new call arrival rate* into cell i is given by $\bar{\lambda}_i = \lambda_i a_i$. Therefore, the expected number of call arrivals during the interval $(0, t]$ is given by $a_i\lambda_i t$. Instead of using the actual value of $\bar{\lambda}_j$ which is not known at the beginning of the new control interval (time 0), each cell i estimates the actual new call arrival rates of its adjacents for the new control period using an exponentially weighted moving average technique, i.e. $\bar{\lambda}_j \leftarrow (1-\epsilon)\frac{N_j}{T} + \epsilon\bar{\lambda}_j$. In our simulations we found that $\epsilon = 0.3$ leads to a good estimation of the actual new call arrival rate.

4.3 Packet Loss Probability

As mentioned earlier, the packet arrival distribution in each cell can be approximated by a Gaussian distribution:

$$R_i(t) \sim \mathbf{G}\Big(E[R_i(t)],\ V[R_i(t)]\Big), \tag{14}$$

where, $E[R_i(t)]$ and $V[R_i(t)]$ are given by (6) and (7), respectively. Using (4) and (14) it is obtained that

$$L_i(t) = \frac{1}{2}\,\mathrm{erfc}\left(\frac{c_i - E[R_i(t)]}{\sqrt{2\,V[R_i(t)]}}\right), \tag{15}$$

where, $\mathrm{erfc}(c) = \frac{2}{\sqrt{\pi}} \int_c^\infty e^{-t^2}\, dt$. Using (15), the average packet loss probability over a control period of length T is given by

$$\tilde{L}_i = \frac{1}{T} \int_0^T L_i(t)\, dt\,. \tag{16}$$

Then, the acceptance ratio, a_i, can be found by numerically solving equation

$$\tilde{L}_i = P_L, \tag{17}$$

where P_L is the target packet loss probability. The boundary condition is that $a_i \in [0,1]$, hence if $\tilde{L}_i$ is less than P_L even for $a_i = 1$ then a_i is set to 1. Similarly, if $\tilde{L}_i$ is greater than P_L even for $a_i = 0$, then a_i is set to 0.

5 Simulation Results

5.1 Simulation Parameters

Simulations were performed on a two-dimensional cellular system consisting of 19 hexagonal cells where opposite sides wrap-around to eliminate the finite size effect. As the basic traffic type, packetized voice calls are generated for simulation purposes. For packetized voice, a packet loss probability of $P_L = 0.01$ is acceptable. The common parameters used in the simulation are as follows. All the cells have the same capacity c. Target packet loss probability is $P_L = 0.01$, control interval is set to $T = 20\,s$ and $r_{ji} = 1/6$. In all of the cases simulated, *normalized load* $\rho = \frac{1}{M_i}(\frac{\lambda}{\mu})$ is used, where M_i is given by (1). For each load, simulations were done by averaging over 8 samples, each for $10^4\,s$ of simulation time. Unless otherwise mentioned, call duration and cell residency times are exponentially distributed with means $\mu^{-1} = 180\,s$ and $h^{-1} = 100\,s$, respectively. We found this set of parameters more or less common and reasonable for a simulation setup (see for example [10]).

A two-state Markov model is used to describe the traffic generation process of voice calls. In this model, α and β are transition rates to OFF and ON states, respectively, from ON and OFF states. While in the ON state, traffic is generated at a constant rate of A packet/sec. For this traffic model, the mean and variance of the traffic generated are given by $E[r] = \frac{\beta}{\alpha+\beta}A$ and $V[r] = \frac{\alpha\beta}{(\alpha+\beta)^2}A^2$. Commonly used parameters for human speech representation are $\alpha^{-1} = 1.2\,s$ and $\beta^{-1} = 1.8\,s$ [12]. Using an 8 Kbps encoded voice source, it is obtained that $A = 100$ packet/sec and hence, $E[r] = 40$ packet/sec and $V[r] = 50$ packet/sec assuming that each packet is 80 bits.

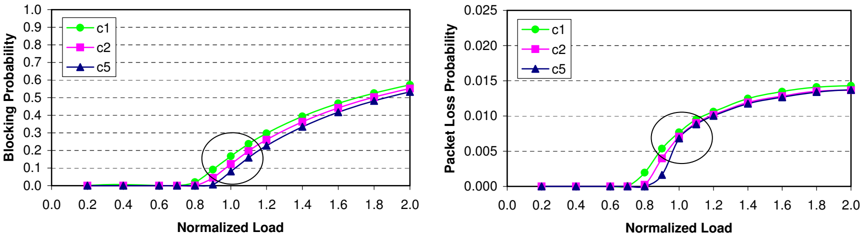
5.2 Conservative PFG

As mentioned earlier, PFG does not drop any handoff call, instead some packets may be dropped to accommodate the incoming handoff packets. To have an intuition about the impact of accepting handoffs even during the overloaded state, we have also implemented a slightly different version of PFG in addition to the original PFG represented in Fig. 1. This modified version drops handoffs during the overloaded state. We refer to the original algorithm by PFG-D0 and the modified one by PFG-DP where D0 and DP stand for zero dropping probability and P dropping probability, i.e. if we use PFG-DP instead of PFG-D0 then there will be P percent call dropping. Our purpose is to find the value of P for some simulated scenarios to see how far it is from zero. Notice that, having $N_i(t) > M_i$ ($t \in (0, T]$) indicates that cell i is in the overloaded state at time t.

5.3 Results and Analysis

As mentioned earlier, PFG is the first to achieve zero call dropping while guaranteeing a hard constraint on packet loss probability. To the best of our knowledge there is no existing scheme which takes into consideration a combination of call-level and packet-level QoS parameters while taking into consideration the *mobility of users*. Therefore, we are not able to compare the performance of PFG with any other scheme. Instead, by doing extensive simulations, we have shown that PFG can achieve its defined goals.

Effect of Cell Capacity: Intuitively, increasing the cell capacity leads to a better Gaussian approximation, and hence, a more accurate admission decision. To investigate the effect of cell capacity, we considered three different capacity configurations, namely, $c1$ = 1 Mbps, $c2$ = 2 Mbps and $c5$ = 5 Mbps (for 3G and 4G systems). Normalized loads in range $[0 \ldots 2]$ are simulated, where the normalized load is defined as mentioned before. In Figs. 3(a) and 3(b) we have circled a region around load $\rho = 1.0$. This is the most interesting part of the system which is likely to happen in practice. In the following discussion we refer to this region as the *operating region* of the system.



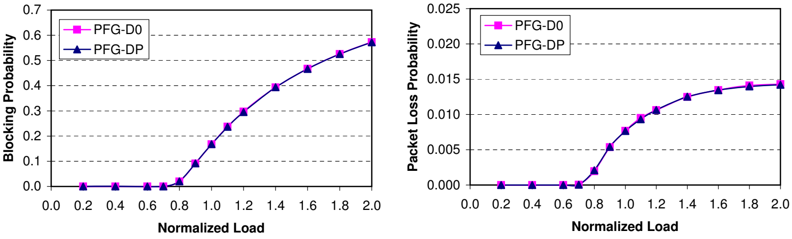
(a) Blocking probability

(b) Packet loss probability

Fig. 3. PFG-D0 performance results

Fig. 3(a) shows the new call blocking probability. It is clear from the figure that as the cell capacity increases the blocking probability decreases which can be explained from the central limit theorem and Gaussian approximation used in section 4.3. The packet loss probability, $\tilde{L}_i$, is depicted in Fig. 3(b). Although $\tilde{L}_i$ goes beyond the target limit for high system loads, it is completely satisfactory for the operating region. Nevertheless, it is quite possible to modify PFG-D0 in order to make it more conservative for high loads. Similar to call blocking, as the capacity increases the PFG-D0 efficiency improves. After all, $c1$ produces rather accurate results and increasing the capacity beyond it produces only marginal improvements.

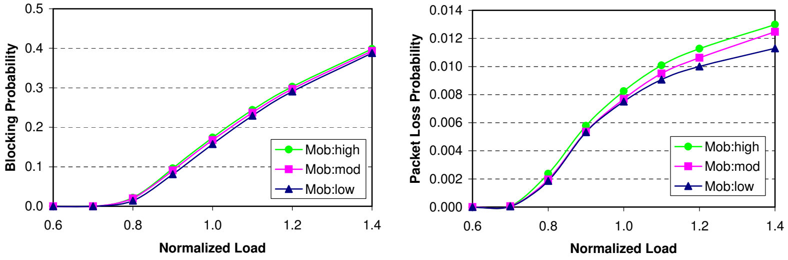
Effect of Accepting Handoffs in Overloaded State: To investigate the impact of accepting handoffs during the overloaded state (in which $N_i(t) > M_i$), we ran PFG-DP for the same simulation configuration we ran PFG-D0. It was observed that the call dropping probability is almost zero in all the simulated configurations (please refer to [13] for detailed numerical results). It means that basically there is no difference between two schemes in terms of the call dropping probability. Fig. 4 shows the call blocking and packet loss probabilities of PFG-D0 versus PFG-DP when the system capacity is set to $c1$ (1 Mbps). It can be seen from Figs. 4(b) and Fig. 4(a) that the call blocking probability and packet loss probability are almost the same for both schemes indicating that accepting handoffs during the overloaded state has a negligible effect on the admission control performance.



(a) Blocking probability (b) Packet loss probability

Fig. 4. PFG-D0 vs. PFG-DP

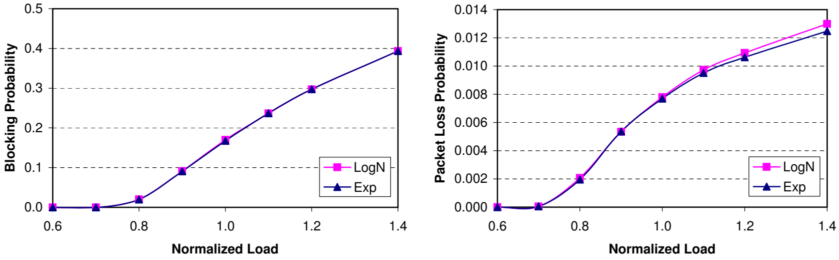
Effect of Mobility: Define the *mobility factor* to be $\alpha = h/\mu$. Intuitively, α shows the average number of handoff attempts a call makes during its life time. As the mobility factor increases the handoff arrival rate increases as well. To investigate the impact of mobility on PFG, we have simulated three mobility cases, namely, (Mob:high, $\alpha = 9.00$), (Mob:mod, $\alpha = 1.80$) and (Mob:low, $\alpha = 0.36$) for the base capacity $c1$. Observed from Fig. 5, PFG is almost insensitive to the mobility rate of users. As shown in Figs. 5(a), the call blocking probability is almost match. Furthermore, Fig. 5(b) shows that the effect of mobility on packet loss probability is not very significant. In all three cases, PFG is able to satisfy the target packet loss probability in the operating region of the system. In general, handoff degrades the performance of cellular systems.



(a) Blocking probability

(b) Packet loss probability

Fig. 5. Mobility impact on PFG-D0 performance



(a) Blocking probability

(b) Packet loss probability

Fig. 6. Lognormal vs. Exponential residence time

Effect of Non-exponential Cell Residence Times: Using real measurements, Jedrzycki and Leung [15] showed that a lognormal distribution is a more accurate model for cell residency time than the exponential distribution. Fig. 6 shows the call blocking and packet loss probability of exponential cell residency versus lognormal cell residency with the same mean and variance. It is observed that the exponential cell residency achieves sufficiently accurate control. In other words, the control algorithm is rather insensitive to this assumption due to its periodic control in which the length of the control interval is much smaller than the mean residency time.

6 Conclusion

In this paper we presented a novel scheme for admission control and hence QoS provisioning for packet-switched cellular systems. In essence, our approach is the natural generalization of the well-known *effective bandwidth* concept proposed for wireline networks. Through analysis and simulation, we showed that the proposed scheme, PFG, is able not only to improve utilization of scarce wireless bandwidth thanks to the statistical multiplexing of VBR traffic sources but also to eliminate the undesirable call dropping event inherent to circuit-switched cellular systems. In wireless multimedia networks, there are different service classes, each of which has its own packet and call level QoS constraints. We are currently investigating the extension of PFG to multiple service classes where each service class has its own QoS requirements.

References

1. Zahariadis, T., Kazakos, D.: (R)evolution toward 4G mobile communication systems. IEEE Wireless Commun. Mag. **10** (2003) 6–7
2. Hui, S.Y., Yeung, K.H.: Challenges in the migration to 4G mobile systems. IEEE Computer **41** (2003) 54–59
3. Peha, J.M., Sutivong, A.: Admission control algorithms for cellular systems. ACM/Kluwer Wireless Networks **7** (2001) 117–125
4. Fang, Y., Zhang, Y.: Call admission control schemes and performance analysis in wireless mobile networks. IEEE Trans. Veh. Technol. **51** (2002) 371–382
5. Gao, Q., Acampora, A.: Performance comparisons of admission control strategies for future wireless networks. In: Proc. IEEE WCNC'02. Volume 1., Orlando, USA (2002) 317–321
6. Hong, D., Rappaport, S.S.: Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedures. IEEE Trans. Veh. Technol. **35** (1986) 77–92 See also: CEAS Tech. Rep. No. 773, College of Engineering and Applied Sciences, State University of New York, June 1999.
7. Ramjee, R., Towsley, D., Nagarajan, R.: On optimal call admission control in cellular networks. ACM/Kluwer Wireless Networks **3** (1997) 29–41
8. Naghshineh, M., Schwartz, M.: Distributed call admission control in mobile/wireless networks. IEEE J. Select. Areas Commun. **14** (1996) 711–717
9. Epstein, B.M., Schwartz, M.: Predictive QoS-based admission control for multiclass traffic in cellular wireless networks. IEEE J. Select. Areas Commun. **18** (2000) 523–534
10. Wu, S., Wong, K.Y.M., Li, B.: A dynamic call admission policy with precision QoS guarantee using stochastic control for mobile wireless networks. IEEE/ACM Trans. Networking **10** (2002) 257–271
11. Ghaderi, M., Boutaba, R.: Mobile effective bandwidth and its application to admission control in cellular packet networks. Technical Report CS-2004-61, School of Computer Science, University of Waterloo (2004)
12. Schwartz, M.: Broadband Integrated Networks. Prentice Hall, New Jersey, USA (1996)
13. Ghaderi, M., Boutaba, R.: Stochastic admission control for quality of service in wireless packet networks. Technical Report CS-2004-60, School of Computer Science, University of Waterloo (2004)
14. Eun, D.Y., Shroff, N.B.: A measurement-analytic approach for QoS estimation in a network based on the dominant time scale. IEEE/ACM Trans. Networking **11** (2003) 222–235
15. Jedrzycki, C., Leung, V.C.M.: Probability distribution of channel holding time in cellular telephone systems. In: Proc. IEEE VTC'96. Volume 1., Atlanta, GA (1996) 247–251

Extending the Birkhoff- von-Neumann Switching Strategy to Multicast Switches

Jay Kumar Sundararajan, Supratim Deb, and Muriel Médard*

{jaykumar, supratim, medard}@mit.edu

Abstract. This paper extends the Birkhoff-von Neumann unicast switching strategy to the multicast case. Using a graph theoretic model we show that the rate region for a traffic pattern is precisely the stable set polytope of the pattern's 'conflict graph', in the no-fanout splitting case. Computing the offline schedule is equivalent to fractional weighted graph coloring which takes polynomial time for perfect graphs. For a general conflict graph, we show that deciding achievability of a given rate vector is NP-hard, but can be done in polynomial time for the case of moderate multicast load. The result naturally leads to an offline schedule.

1 Introduction

Online unicast scheduling in input-queued switches is a well-studied problem. McKeown et al. [1] gave a scheduling algorithm known as the maximum weighted matching (MWM) algorithm which achieves 100 % throughput for all admissible unicast arrival patterns. One major drawback of MWM-based approaches is that they do not provide cell delay guarantees. The Birkhoff-von Neumann (BVN) switch proposed by Chang et al. [2] addresses this issue. The BVN switch provides 100 % throughput for all non-uniform traffic, and also gives deterministic cell delay guarantees for certain types of traffic.

The basic idea of the BVN approach is as follows. Suppose the average arrival rate of packets is known for every input-output pair. This rate requirement matrix is decomposed into a convex combination of permutation matrices. Since a permutation matrix naturally corresponds to a switch configuration, this decomposition leads to an offline[1] schedule. In this paper, we study the extension of this approach to the case when there are multicast flows within the switch.

A fundamental question that arises while trying to extend the BVN strategy to multicast is that of how to split the *fanout* (the set of destinations) of each multicast flow. There are many options each requiring a different queuing policy:

* The authors are with Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

[1] In our work, 'online' means decide the switch configuration for each slot based on the current queue occupancies, whereas 'offline' methods assume knowledge of the average rates for each flow, and decompose the rate vector into a sequence of switch states.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1321–1325, 2005.

1. **Copying:** Each input maintains one virtual output queue (VOQ) for every output. When a multicast cell arrives at an input, it is replicated as many times as its fanout size. One copy is added to the VOQ of each of the outputs in the fanout. This is equivalent to viewing the multicast as a collection of unicast flows.
2. **No-splitting:** This is the other extreme, where a multicast cell is sent to all outputs in its fanout in a single time slot. Each input must maintain a separate VOQ for every multicast flow. Moreover, the switch should have *intrinsic multicast capability*[2].
3. **Fanout-splitting:** Multiple copies of the multicast cell are generated and, in each slot, one copy is transferred to a subset of the fanout which has not already got the cell. The fanout is thus split partially. Again, the switch needs to have intrinsic multicast capability. This can be implemented in two ways:
   - **Static splitting:** Here, the multicast traffic pattern is assumed to be known before hand. All cells of a flow are split in the same manner (which is decided offline). This is like replacing the original multicast with a set of "split flows", for which further splitting is not allowed. Each input must maintain a separate VOQ for every split flow. When a multicast cell arrives at an input, it is replicated according to the predecided policy, and one copy is added to the VOQ of each of its split flows.
   - **Dynamic splitting:** For this strategy, each input maintains a VOQ for every subset of the fanout. When a multicast cell arrives at an input, it is transferred to some subset of its fanout and then, re-enqued into the VOQ corresponding to the remaining part of the fanout. The way in which a multicast flow's fanout is split can vary from cell to cell.

Dynamic splitting is the least constrained approach and subsumes the other options. It is known that, in the online case, dynamic fanout-splitting gives better throughput than no-splitting [3]. However, this benefit comes at a cost. Since the way flows are split is not known beforehand, the part of the fanout that remains to be served at some point of time, could be any arbitrary subset of the original fanout. Therefore, each input has to maintain a separate VOQ for every possible subset of the fanout, resulting in an exponential number of queues, even if the traffic pattern is known beforehand. In contrast, *the static splitting approach requires a much smaller number of queues.* At each input, one VOQ for every split flow is enough.

Earlier work on multicast switching has focused on online algorithms when dynamic splitting is allowed. Marsan et al. [4] defined the optimal online multicast scheduling algorithm, and defined the capacity region, but did not give an explicit characterization.

In this paper, we present a graph theoretic model for the problem, and use it to derive the rate region of the no-splitting case. We also present an algorithm

[2] The ability to simultaneously transfer a cell to multiple outputs using simultaneous switching paths.

to decide the achievability of a given set of rates, and find an offline schedule, in the case of moderate multicast load.

2 Rate Region and Offline Schedule

Definition 1 (Conflict Graph). *The conflict graph $G = (V, E)$ for a given traffic pattern is defined thus:*

$$V = \textit{set of all flows to be served}$$
$$E = \{(v_i, v_j) | \textit{flows } i,\ j \textit{ cannot coexist within a valid switch configuration}\}.$$

The conflict graph brings out the connection constraints in the switch. A valid switch configuration consists of a set of flows that can co-exist, which corresponds to a set of vertices no two of which are connected - in other words a *stable set*. A convex combination of stable sets gives a schedule where the flows in a particular stable set are served for a fraction of time equal to the coefficient in the combination. This leads to the following result. (The proof of the theorems in this paper are not given here for want of space, and can be found in [6]).

Theorem 1. *The rate region for the multicast case with no-splitting is precisely the stable set polytope of the conflict graph.* □

Next, we address the problem of computing the offline schedule in a general case. This approach gives an efficient algorithm for the case of perfect conflict graphs. [6] gives several examples of traffic flows whose conflict graph is perfect.

Theorem 2. *The problem of computing the decomposition of the given rate requirements into switch connection states (stable sets) is precisely the problem of* fractional weighted graph coloring *of the conflict graph, with the weights chosen as the required rates for the flows.* □

3 Deciding Achievability

Theorem 3. *The problem of deciding whether a given rate requirement vector is achievable using the no-splitting strategy is NP-hard.* □

The basic idea of the proof is to map the decidability question to the membership problem over the stable set polytope of a general graph. We next show that the hardness result can be extended to the case when dynamic fanout splitting is allowed. For details of the proof, please refer to [6].

Theorem 4. *The problem of deciding whether a given rate vector is within the rate region of the no-splitting case can be reduced to an instance of the corresponding problem in the fanout-splitting case.* □

Corollary 1. *The problem of deciding achievability in a multicast switch when fanout-splitting is allowed, is NP-hard.* □

We now present an algorithm to decide the achievability of a given rate vector, with no fanout-splitting. The algorithm runs in time polynomial in the switch size (N) if the number of multicasts is $O(logN)$(see [6] for details). This algorithm naturally gives a schedule to achieve the rates in a stable manner.

Algorithm 1 DECIDER:
INPUT: A rate requirement vector $\mathbf{r_o}$*; a traffic pattern in an* $N \times N$ *switch, with* k *multicasts and all possible unicasts.*
OUTPUT: Is $\mathbf{r_o}$ *within the achievable region corresponding to the traffic pattern, in the no-splitting case? If yes, give a schedule to achieve it in a stable manner.*

1. *Let the multicasts be* $M_1, M_2, \ldots, M_k$*. Let* $A \subseteq [k]$*. Let* R_A *be the rate region for the traffic pattern with the condition that the multicasts* $\{M_i | i \in A\}$ *are all always being served. Compute* R_A *for all possible subsets* A *of* $[k]$*.*
2. *Verify whether* $\mathbf{r_o}$ *lies in the convex hull of all the* R_A*'s. The answer to this question is the output.*
   Let $\mathbf{B_j x} \leq \mathbf{c_j}, j = 1, 2, \ldots, J$ *be the set of convex regions representing* R_A*. Verifying whether a point* $\mathbf{r}$ *is in the convex hull of these regions is equivalent to verifying whether this linear program in the variables* $\mathbf{y_j}$ *and* ϕ_j *is feasible:*

$$\sum_{j=1}^{J} \phi_j = 1; \quad \phi_j \geq 0; \quad \mathbf{r} = \sum_{j=1}^{J} \mathbf{y_j}; \quad \mathbf{B_j y_j} \leq \phi_j \mathbf{c_j} \quad \forall j = 1 \text{ to } J. \quad (1)$$

4 Conclusions

In this paper, we have investigated offline multicast switch scheduling using a graph theoretic formulation of the problem in terms of conflict graphs. We have shown that the rate region for a given traffic pattern is the stable set polytope of the conflict graph. We have provided an algorithm to decide the achievability of a rate vector and find the offline schedule for a moderate number of multicasts, in the no splitting case. Note that static splitting can be seen as an instance of no-splitting, with the set of flows chosen as the split flows. So, all results derived for no-splitting strategy in this paper hold for the static splitting strategy, which is very attractive as it requires a much smaller number of queues than dynamic splitting. The i-SLIP [5] unicast algorithm can be extended to multicast using the conflict graph formulation presented here. This is explained in detail in [6].

References

1. N. McKeown, V. Anantharam, J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch", *Proc. of IEEE INFOCOM '96*, pp. 296-302.
2. C.S. Chang, W.J. Chen and H.Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," *Proceedings of IEEE INFOCOM 2000*, pp. 1614-1623, Tel-Aviv, Israel, March 2000.

3. N. McKeown, “A Fast Switched Backplane for a Gigabit Switched Router”, *Business Comm. Review*, Vol. 27, No. 12, Dec. 1997
4. M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri: “Multicast traffic in input-queued switches: optimal scheduling and maximum throughput.” *IEEE/ACM Trans. on Networking* Vol. 11, No. 3, June 2003, pp. 465-477.
5. N. McKeown, “The i-SLIP scheduling algorithm for input-queued switches”, *IEEE/ACM Transactions on Networking*, Vol. 7, no. 2, Apr. 1999.
6. Jay Kumar Sundararajan, “Extending the Birkhoff-von Neumann Switching Strategy to Multicast Switching”, S.M. Thesis, MIT, January 2005

A New TCP-Friendly Rate Control Algorithm for Scalable Video Streams

Jinyao Yan[1,2], Martin May[1], Kostas Katrinis[1], and Bernhard Plattner[1]

[1] Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology,
ETH Zurich, 8092, Switzerland
[2] Communication University of China, Beijing, 100024,China
{jinyao,maym, katrinis,plattner}@tik.ee.ethz.ch

Abstract. This paper presents Media- and TCP-Friendly Rate Control Algorithm (i.e. MTFRC) for scalable video streams. We develop a two-timescale approach to satisfy media friendliness in the short-term, while maintaining TCP friendliness in the long-term. We evaluate MTFRC with five criteria, namely TCP-fairness, responsiveness, aggressiveness, video quality and smoothness of sending rate. Our simulation results manifest that MTFRC outperforms the TCP-Friendly Rate Control Protocol (TFRC) and improves the overall video quality.

1 Introduction

TCP uses an additive increase multiplicative decrease mechanism (AIMD) to detect additional available bandwidth and to react to congestion in the steady state. However, it is not well suited for the growing number of media streaming applications. As a result, new TCP-friendly congestion control protocols such as TFRC [1] have been proposed for streaming applications while competing TCP flows friendly. Still, there is a fundamental shortcoming of the above protocols: they target at optimizing the utilization of network resources *without taking the resulting application quality into account*. Even though TFRC does smooth the rate variability, this does not guarantee maximum media quality at the users.

Given a system with scalable video streaming sources/receivers and links, the challenge is to allocate a rate value to each source that is both media- and TCP-friendly. A rate control algorithm is termed media-friendly, if it optimizes the overall quality of the various video streams in the network.

The main contribution of this paper is the MTFRC algorithm that optimizes the overall video quality for scalable streams, while being TCP-friendly in the long term.

2 Transforming MTFRC into a TCP-Friendly Algorithm

In our previous work [2], we introduced the video quality function for scalable video as the utility function in a utility-based network model and used it to maximize video

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1326 – 1329, 2005.

quality. Let $I_s = [m_s, M_s]$ denote the feasible range of rate x_s for a scalable video source s. We derived the optimal throughput response function $x_s(p)$ that maximizes the overall video quality of scalable video streams (a, b and c are parameters to be evaluated empirically by the video stream quality):

$$x_s(p) = [(\sqrt{\log_2^{(-p/a\ln 2)}/a - c/a + b^2/4a^2} - b/2a)^2]_{m_s}^{M_s} \tag{1}$$

To behave in a TCP-friendly manner, the TFRC protocol adjusts its sending rate based on the TCP throughput function in [3].

Let the throughput response function of MTFRC Eq. (1) be denoted by $X_{MTFRC}(p)$; the throughput response function of TFRC be denoted by $X_{TFRC}(p)$. When two traffic flows compete for resources at the same network bottleneck, both flows will obtain a fair share of the bandwidth, if and only if both flows apply similar long-term response functions. In order to meet TCP friendliness, we modify the MTFRC algorithm in a two-timescale approach as follows:

1. The sender (or receiver) estimates two congestion control parameters: the long-term packet loss rate p_L, and the short-term packet loss rate p_S.
2. The sender calculates its long-term rate using the long-term packet loss rate p_L according to the TFRC response function. However, the sender reacts to short-term congestion given by p_S according to the MTFRC response function (1).

In particular, the MTFRC algorithm adjusts the parameters a, b, c in Eq. (1) to achieve the long-term rate $X_{TFRC}(p_L) = X_{MTFRC}(p_L)$, the sending rate for next period is $X_{MTFRC}(p_S)$ using updated values for a, b, or c.

3 Results and Analysis

In this section, we present the results obtained from simulations we conducted to evaluate the behavior of the MTFRC algorithm against TFRC. We used the ns2 network simulator [5] for our experiments. The scenario implemented throughout our simulations is illustrated in Figure 1. Physical links are marked with propagation delay and capacity bandwidth values (default is 10Mbps/10ms). We considered the following evaluation criteria: 1) TCP-friendliness of competing flows 2) Responsiveness: reaction time of the protocol, when decreasing the sending rate in case of severe network congestion 3) Aggressiveness: acceleration of protocol sending rate after a congestion incidence 4) Smoothness: rate variations over time for a particular flow 5) Overall video quality.

We compare the characteristics of our MTFRC algorithm to those of TFRC(8) (history size 8) and TFRC(128) (history size 128). For MTFRC, we used a long-term history size of N_L=256 and a short-term history size of Ns=8 (denoted as MTFRC(256, 8)).

Figure 2, Fig. 3, and Fig. 4 present the bit rate evolution of the two protocols in the event of severe congestion and sudden increase of the available bandwidth. We use a CBR source (sending rate: 1Mbps) on Conn.4 to decrease the available bandwidth at the 60th second (severe network congestion) and to increase the available bandwidth at the 120th second.

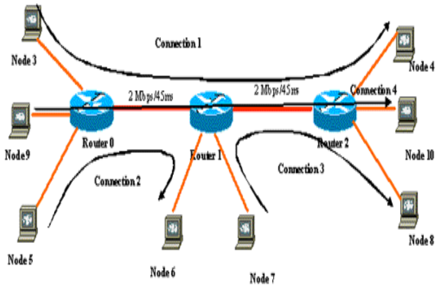


Fig. 1. Network Topology

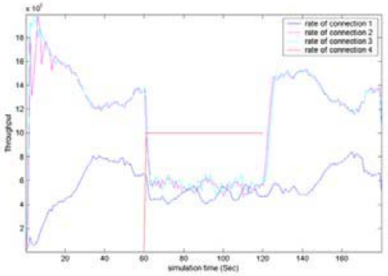


Fig. 2. Bit rate traces with TFRC(8)

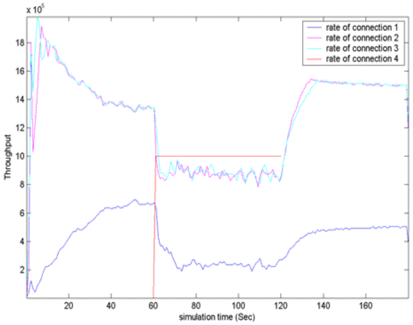


Fig. 3. Bit rate traces with MTFRC(256,9)

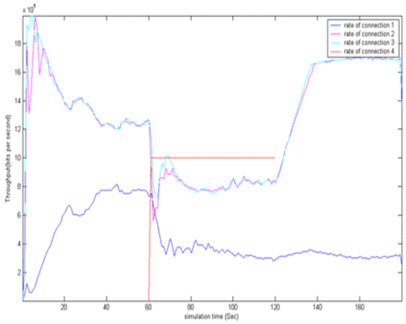


Fig. 4. Bit rate traces with TFRC(128)

Overall Video Quality. We apply the obtained sending rate traces of the simulated network connections to FGS video scaling [4] and obtain the video quality (luminance component) that the MTFRC flow would exhibit. The PSNR (Peak Signal-to-Noise Ratio) values of MTFRC and those achieved by TFRC for the "Highway" test stream are depicted in Table 1.

Due to space limitation, we omit the detailed evaluation of TCP-friendliness, responsiveness, aggressiveness, and smoothness for the MTFRC algorithm. In table 2, we summarize the findings of our evaluation. Overall, MTFRC algorithm offers a better tradeoff between network utilization and media quality.

4 Conclusion and Future Work

In this paper, we studied a both Media- and TCP-friendly rate control algorithm for scalable video streams. We introduced a two-timescale approach of rate averages (long-term and short-term) to the MTFRC algorithm to satisfy both media and TCP friendliness. Using simulations, we showed that our algorithm is TCP-friendly and achieves an improved network-friendly behavior (evaluated in terms of responsiveness, aggressiveness and fairness). Additionally, MTFRC exhibited superior media friendliness

Table 1. Comparison of resulting quality

	Stream on conn.1	Stream on conn.2	Stream on conn.3
PSNR and its variance with MTFRC	27.9 σ = 0.020	32.35 σ = 0.0099	32.36 σ = 0.010
PSNR and its variance with TFRC	29.39 σ = 0.018	30.70 σ = 0.020	30.69 σ = 0.020
Total quality gain	Increase of PSNR over MTFRC=1.8, Decrease of its variance=0.018		

Table 2. Summary of the simulation results

	MTFRC(256,8)	TFRC(8)	TFRC(128)
TCP Fairness	All three algorithms are TCP-friendly		
Smoothness of bit rate	Smoother than TFRC(8), roughly same as TFRC (128)	Smooth bit rate	Smoother than TFRC(8) similar to MTFRC
Responsiveness	Medium	High	Low
Aggressiveness	Faster than TFRC (128), slower than TFRC(8)	Faster than TFRC (128) and MTFRC	Slower than TFRC(8) and MTFRC
Video quality	Better quality than TFRC (8) and TFRC(128)	Similar quality to TFRC(128), worse than MTFRC	Similar quality to TFRC(8), worse than MTFRC

than that of TFRC in terms of smoothness and video quality. Our ongoing work will focus on investigating the behavior of MTFRC algorithm in the open Internet.

References

1. S. Floyd, M. Handley, and J. Padhye, “Equation-Based Congestion Control for Unicast Applications”, ACM SIGCOMM, September 2000.
2. Jinyao Yan, Kostas Katrinis, Martin May, Bernhard Plattner, “Optimizing Rate Control for Multiple Fine-granular Scalable Video Streams”(short paper), IEEE ICNP 2004, Berlin, Germany Oct. 2004.
3. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. “Modeling TCP Throughput: A Simple Model and its Empirical Validation”. SIGCOMM Symposium on Communications Architectures and Protocols, Aug. 1998.
4. W. Li, “Overview of Fine Granularity Scalability in MPEG-4 Video Standard”, IEEE Trans.on Circuits and Systems for Video Technology, Vol.11, No.3, March 2001, pp301-317
5. ns-2 network simulator, http://www.isi.edu/nsnam/ns/

802.11 Link Interference: A Simple Model and a Performance Enhancement

Hoon Chang and Vishal Misra

Department of Computer Science, Columbia University, New York NY 10027, USA
{hchang, misra}@cs.columbia.edu

Abstract. Recent findings have revealed that the carrier-sensing range set in current major implementations does not detect all interference signals in 802.11 networks. In this paper, we investigate the effect of interference and develop a mathematical model for it. The accuracy of our model is verified via simulations. Based on an insight from our model, we present a simple protocol that operates on the top of 802.11 MAC layer and achieves more throughput than rate-adjustment schemes.

1 Introduction

It has been reported that IEEE 802.11 stations out of the carrier-sensing range of senders may interfere with receiver stations [5]. Consider two 802.11 senders, stations A and B that can not sense signals from each other but can interfere. During backoff times, two senders do not sense any active signals. They finally start their transmissions, which may overlap and collide. After collisions, the binary exponential backoff in 802.11 DCF grows their backoff windows up until one sender can send a packet without interference.

However, a binary exponential backoff does not provide fairness for a short term [3]. Assume sender B has successfully finished its transmission. A larger backoff window of sender A may allow sender B to capture the channel and Sender A's window size finally reaches the maximum. Note that packet corruption probability is not exactly 1, and sender A can eventually succeed and returns in early backoff stages. Their contention sends one of the senders to the maximum backoff window size again. Our simulation results support this observation and are the basis for our modeling approximation.

In the next section, we present the operational model and analysis on the behavior of 802.11 DCF under interference with two pairs of greedy senders and receivers. Section 3 presents our protocol to provide throughput enhancements. We finally conclude in section 4.

2 Modeling and Analysis

2.1 Operation Model and Upper Bound of Throughput

From our observation, we assume that after every successful transmission, a sender that has finished transmitting goes back to the first backoff stage while

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1330–1333, 2005.

the other sender moves to the last regardless of its current stage. In other words, one sender has the minimum backoff window (CW_{min}) and the other has the maximum (CW_{max}). If both senders make successful transmissions, then one sender is arbitrarily selected to go to the first stage.

The operational model has m states based on our analysis assumption. Each state corresponds to a pair of backoff stages of two senders in the network. In state (i, m), one of senders stay in stage i and the other in the last stage m. In 802.11, their window sizes are $(CW_{min} + 1) \cdot 2^{(i-1)} - 1$ and CW_{max}. If a collision occurs, two senders move to the next state $(\min(i + 1, m), m)$ and increase their window size. If one of them makes a successful transmission, the senders go back to $(1, m)$. Analysis on this assumption can provide an upper bound of the channel throughput because setting two senders in the first and the last stages minimizes the conflict probability [1].

2.2 Average Transmission Time for a Single Packet

We present in this paper analysis for only basic accesses due to space limitations. Complete analysis for RTS/CTS accesses is found in [1].

Define very small virtual slots and assume that a sender in stage i starts a transmission at the beginning of a virtual slot with probability $\lambda_i = 1/((W_i - 1)/2 \times T_{SLOT} + T_{DIFS})$, where T_{SLOT} and T_{DIFS} are the length of 802.11 slots and DIFS in unit of virtual slots. At each slot, transmission probability τ_i of two senders in state (i, m) is $\lambda_i + \lambda_m - \lambda_i\lambda_m$. Let total transmission time of a B-bit packet be T_{TX}, which includes time to send a data packet (T_{DA}) and an ACK. For the average collision time, we assume that the second transmission starts at the beginning of any virtual slot within T_{DA} and it follows the uniform distribution. Let T_{OVER} be the average.

Given λ_i and λ_m, the probability p where a transmission is completed without overlapping is $(\lambda_i \cdot (1 - \lambda_m)^{T_{DA}} + \lambda_m \cdot (1 - \lambda_i)^{T_{DA}})/\tau_i$. Suppose that at least one sender begins a transmission in state (i, m) and let p_i be the conditional probability of successful transmissions occurring. We obtain $p_i = p + (1-p) \times (1 - P_{PER}^2)$, where P_{PER} is error rate of an overlapped packet. Note that $P_{PER} \leq 1$ and overlapped transmissions result in safe deliveries with probability $(1 - P_{PER}^2)$. Let q be the average number of packets that are overlapped but delivered.

Now we obtain C_j, which is time to reach stage (j, m). C_1 is equal to 0.

$$C_m = C_{m-1} + 1/\tau_{m-1} + T_{OVER} + (1/\tau_m + T_{OVER}) \times (1 - p_m)/p_m \qquad (1)$$

$$C_i = C_{i-1} + 1/\tau_{i-1} + T_{OVER} \quad (2 \leq i \leq m - 1) \qquad (2)$$

Note that senders must stay in the same state after conflicts in (m, m), which happens $(1/p_m - 1)$ times on average. Now consider the average delivery time S_i of one packet in state (i, m). To reach the state (i, m) from $(1, m)$ takes C_i and successful transmissions in (i, m) happen with probability p_i. Assuming successful transmissions, a successful transmission without overlapping occurs at p/p_i and takes $1/\tau_i + T_{TX}$. With overlapping, q packets are delivered in time $1/\tau_i + T_{OVER}$. Thus, S_i is:

$$S_i = p/p_i(C_i + 1/\tau_i + T_{TX}) + (1 - p)/p_i(C_i + 1/\tau_i + T_{OVER})/q \qquad (3)$$

Now, let T be the average delivery time of one packet. We obtain T from S_i as follows and the throughput in this system is B/T_0.

$$\begin{aligned} T &= S_1 \times p_1 + (1 - p_1) \times [S_2 \times p_2 + (1 - p_2) \times (\ldots)] \\ &= S_1 p_1 + \sum_{i=2}^{m-1} S_i p_i \prod_{j=1}^{i-1} (1 - p_j) + S_m \prod_{j=1}^{m-1} (1 - p_j). \end{aligned} \tag{4}$$

2.3 Simulation in `Qualnet`

In `Qualnet` [4], we used two-ray path loss model. From two-ray path loss model and BER mapping function in `Qualnet`, we can compute the acceptable range of SINR values and the interference range. 802.11 stations were placed at locations where packet loss probabilities for simulations can be achieved. All stations were stationary with omni-directional antennas and located on a flat plain without any obstacles. Fading models were not used.

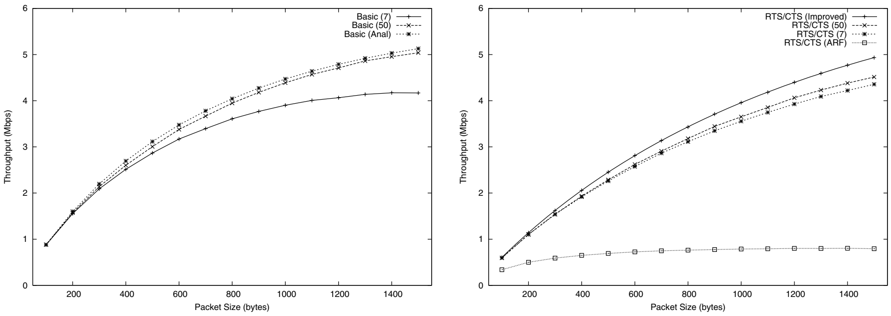
Figure 1(a) shows the simulation results in terms of the cumulative channel throughput with varying packet size. Simulation result for RTS/CTS accesses is presented in [1]. Two senders and receivers transmitted data packets and ACKs at 11 Mbps. Each simulation, which runs with fixed packet size in the range of 100 to 1500 bytes, was performed for 100 seconds. We also varied the long and short retry numbers in 802.11; the graph tagged 'BASIC (50)' shows results setting both of long and short retry limits to 50. The operation with default retry numbers are also plotted in the graph 'BASIC (7)'.

As expected in section 2, analysis results put the upper bounds on the channel throughput, that look very tight. By increasing retry numbers up to 50, the channel throughput with sending 1500-byte packets hit 98.28 percent of the expectation. This indicates that setting large retry numbers keeps one of senders in the last backoff stage longer than in the regular cases and it boosts up the chance to make a successful transmission.

3 Throughput Enhancement

The key idea of this enhancement protocol is emulating the analysis model to reduce conflicts and guarantee bounded waiting time as much as possible. When the protocol detects a packet loss, it delays supplying packets to the MAC layer for a random backoff time in the range of 400 to 2000 slots. After that, it tries to (re)transmit a packet until the packet delivered or it moves back to the delay state after a timeout. If it has delivered at least one packet and detects a loss again, it transits to the delay state. Full state diagram is found in [1].

Figure 1(b) shows performance of our protocol when RTS/CTS probing is used. Simulation result for basic accesses is present in [1]. As expected, our protocol enhanced the system throughput more than simple increase of retry numbers. We also run Lucent AutoRate Fallback (ARF) protocol [2] and denote the results *ARF* in the figure. Due to long consecutive losses, senders in ARF reduce the transmission rate until it reaches 1 Mbps, where transmissions are



(a) Throughput Results (Basic)

(b) Improved Results (RTS/CTS)

Fig. 1. Simulation Results at 11 Mbps

more vulnerable. Senders, thus, do not increase their transmission rates while our protocol effectively reduces conflicts and enhances the throughput.

4 Conclusion

In this paper, we investigated the behavior of 802.11 DCF on the interference channel. We presented our analysis model and performed simulations that show our analysis provides tight upper bounds. According to our analysis, delaying packets from the upper layer will mitigate the impact of packet conflicts. We presented a simple protocol that increases the system throughput by 30.9% while Lucent's ARF protocol, one of well-known rate adaptation schemes achieves much smaller than regular 802.11 DCF throughput without rate adjustment.

References

1. Hoon Chang and Vishal Misra. 802.11b throughput with link interference. Technical report, Columbia University, 2004.
2. A. Kamerman and L. Monteban. WaveLAN-II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, pages 118–133, 1997.
3. K. K. Ramakrishnan and Henry Yang. The ethernet capture effect: Analysis and solution. In *the IEEE 19th Local Computer Networks*, 1994.
4. Scalable Network Technologies. QualNet Developer, 2004. `http://www.qualnet.com`.
5. Kaixin Xu, Mario Gerla, and Sang Bae. How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks? In *IEEE Globecom*, 2002.

Impact of Routing Lags on Internet Routing Failures[⋆]

Feng Wang and Lixin Gao

Department of Electrical and Computer Engineering,
University of Massachusetts, Amherst, MA 01002, USA
{fewang,lgao}@ecs.umass.edu

Abstract. Studies have shown that the Internet has experienced the widespread failures such as router crash, fiber cut, or scheduled maintenance. Ideally, routing protocols should be able to quickly find alternate paths to reroute around failures. In this paper, we consider one important factor that prevents routing protocols from achieving this goal: the delay of finding and obtaining alternate paths, defined as *routing lag*. We show that a significant number of routing failures in the Internet are caused by routing lags, and they can last for a significant period of time.

1 Introduction

One of the goals of routing protocols is to be able to reroute around failures such as router crash, fiber cut, or scheduled maintenance. When a failure occurs, routing protocols should be able to quickly find alternate paths to reroute around failures. Routing protocols failing to achieve this goal is called as *routing failure*. Widespread routing failures in the Internet have been observed in experimental studies [1, 2]. Furthermore, Border Gateway Protocol (BGP), a path vector protocol for interdomain routing in the Internet, experiences considerable delay in reaching convergence [3, 4, 5]. BGP can experience transient loss of reachability during route convergence. For example, BGP applies "poison reverse" to avoid routing loops. If router A uses router B to reach a destination, router A does not announce its best route to router B. This can limit the route visibility of router B so that there is latency of obtaining an alternate route from A during the path exploration. We define the delay of finding and obtaining alternate routes as *routing lag*.

In this paper, we study routing failures due to routing lags in the Internet. First, we show that routing lags are prevalent in the sense that any router can experience routing lags. Second, we derive upper bound for routing failure durations due to routing lags, and consider the impact of rate limiting timers on the duration. We find that if the rate limiting timers are applied on the granularity of a peer (current implementation) instead of a prefix, routing lags can last for a significant period of time. Our results imply that there is a clear need to reconsider the practice of rate limiting timers.

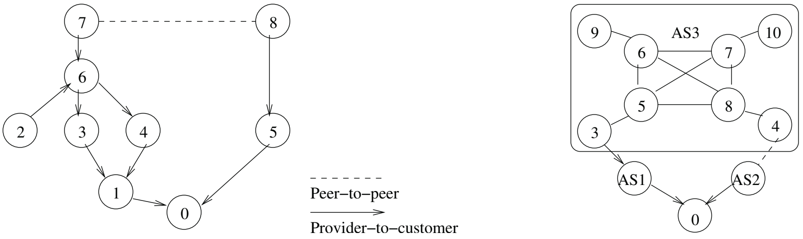
[⋆] This work was supported by the NSF under NSF CNS-0208116 and NSF CNS 0325868. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1334–1338, 2005.

2 Routing Lags in the Internet

We study routing lags in a typical BGP system, which means that every router in the system applies *common routing policies*. Routing policies are guided by commercial relationships between ASes: *customer-to-provider* and *peer-to-peer*. In a typical BGP system, the import routing policies are guided by the *prefer-customer* routing policies, while export routing policies by the *no-valley* routing policies.

Routing lags can be prevalent in a typical BGP system. Figure 1(a) shows a typical BGP system, in which we assume that each AS contains only one router and the destination is situated in AS 0. Suppose that the link between nodes 1 and 0 breaks, node 1, 3, 4 and 6 will definitely, and node 2 will possibly experience routing lags. On the other hand, large ASes typically have a hierarchical iBGP structure. For example, in Figure 1(b), AS 3 has the customer AS 1 and the peer AS 2 so that all routers inside AS 3 will use the path via router 3 to reach the destination in AS 0. Once the link between router 3 and AS 1 fails, all routers inside the AS except router 4 may experience routing lag.



(a) Routing lags involving multiple ASes (b) Routing lags inside an AS

Fig. 1. Routing lags take place in a typical BGP system

3 Routing Failure Duration Induced by Routing Lags

Routing failure duration induced by a routing lag is the time interval between the time when there is no route to a destination and the time when the first route appears after the failure. The major factor determining the failure duration is the rate limiting timer, MinRouteAdver (MRAI), which is used to determine the minimum amount of time that must elapse between routing updates. There are two ways to apply rate limiting timers: on the granularity of a prefix and on a peer. For each way, rate limiting timers can be applied to only announcements or both announcements and withdrawals. When the timer is applied to withdrawals (announcements), it is started again after finishing sending the withdrawal (announcement) messages. Next, we will derive the upper bounds of routing failure duration, which is caused by routing lags under different MRAI implementations.

We use an AS graph $G = (V, E, d)$ to represent a network connectivity to a destination d. The node set V consists of routers, and the edge set E represents the connectivity to the destination d. Router u has a set of path $P_{(u,l)} = \{P_1, P_2, \cdots, P_r\}$ to the destination. Suppose a link fails, it will lead router u to reroute the failure. It will request other routers provide alternate paths if they have. We denote these nodes where node u

can obtain a path with a set $\mathrm{B} = \{\beta_1, \beta_2, \ldots, \beta_\mathrm{m}\}$. So failure duration due to routing lags consists of (1) the delay of propagating the request (withdrawal message) from u to any router in B, denoted as $d_{(u,\beta)}$, and (2) the delay of obtaining an alternate path (announcement message) from the router, denoted as $d_{(\beta,u)}$.

Given two neighboring nodes u and v, the MRAI timer of u configured for v is denoted by $M_{(u,v)}$. Obviously, the delay for a routing update from u to v is $d_{(u,v)} \leq M_{(u,v)}$. Note that the value of M can be either for an iBGP or for an eBGP session.

With MRAI timer based on per peer implementation, the upper bound of failure duration is shown as follows:

- If MRAI timer is applied to both announcement and withdrawal, the failure duration due to routing lag is bounded by

$$\min_{\beta \in B}(d_{(u,\beta)} + d_{(\beta,u)}) \leq \min_{\beta \in B}(\sum_{(i,j) \in P_{(u,\beta)}} M_{(i,j)} + \sum_{(i,j) \in P_{(\beta,u)}} M_{(i,j)}) \tag{1}$$

- If MRAI timer is applied to announcement only, the failure duration due to routing lag is bounded by

$$\min_{\beta \in B}(d_{(u,\beta)} + d_{(\beta,u)}) \leq \min_{\beta \in B}(\sum_{(i,j) \in P_{(\beta,u)}} M_{(i,j)}) \tag{2}$$

With MRAI timer based on per prefix implementation, the upper bound of failure duration is shown as follows:

- If MRAI timer is applied to both announcement and withdrawal, the failure duration due to routing lag is bounded by

$$\min_{\beta \in B}(d_{(u,\beta)} + d_{(\beta,u)}) \leq M \tag{3}$$

  In this case, we assume that the first update can be propagated *without delay* imposed by those timers.
- If MRAI timer is applied only to announcement, the failure duration due to routing lag is bounded by

$$\min_{\beta \in B}(d_{(u,\beta)} + d_{(\beta,u)}) \leq \delta, \tag{4}$$

  where δ represents the transmission delay of routing updates at each link and process delay at each router, without any delay imposed by MRAI timer.

We find that applying rate limiting timer on the granularity of a peer can indeed prolong the failure duration compared with on the granularity of a prefix. We also find that when MRAI timer is applied to just announcement but based on per prefix, the duration is the smallest. Further details on the proof of failure duration are presented in [6].

4 Measurement Results

We first measure the prevalence of routing failures due to routing lags at 5 tier-1 ASes by using BGP updates in August 2003 from Oregon RouteView server. The percentage of routing failures at those ASes is shown as following: AS1239 (4.4%), AS2914 (18%), AS3356 (3%), 3561 (2.8%), and AS7018 (8.7%). Figure 2 shows the cumulative distribution of failure duration due to routing lags at 5 tier-1 ASe, which obtain alternate paths from neighboring ASes, in August, 2003. We find that more than 95% of those failures have less than 90 seconds of failure duration, and more than 90% of those failures last more than one MRAI time (30 seconds). Details on the method of identifying routing lags are presented in [6].

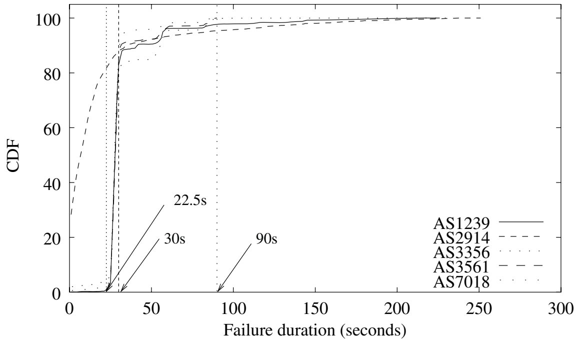


Fig. 2. Distribution of the duration of routing failures due to routing lags

5 Conclusions

In this paper, we analyze the impact of routing lags on routing failures. Our results show that routing lags are prevalent in the Internet, and can indeed lead to longer failures. We also consider the impact of the implementation of rate limiting timer on routing lags. We find that if rate limiting timers are applied on the granularity of a peer instead of a prefix, the failure duration is the longest. Our results imply that there is a clear need to reconsider the practice of rate limiting timers.

References

1. C. Labovitz, G. Malan, F.Jahanian. Internet Routing Instability, In *Proc.* of SIGCOMM, September 1997.
2. C. Labovitz, A. Ahuja, F. Jahanian, Experimental Study of Internet Stability and Backbone Failures. FTCS 1999: 278-285.
3. C. Labovitz, A. Ahuja, and et al, Delayed Internet Routing Convergence. IEEE/ACM trans. On Networking, Vol.9, No.3, June 2001.

4. C. Labovitz, A. Ahuja, The Impact of Internet Policy and Topology on Delayed Routing Convergence. In *Proc.* of INFOCOM 2001, Anchorage, Alasks, April 2001.
5. T. Griffin and B. J. Premore, An experimental analysis of BGP convergence time. In *Proc.* ICNP 2001, Nov 2001.
6. Feng Wang, Lixin Gao, Impact of Routing Lags on Internet Routing Failures, Techincal report, http://rio.ecs.umass.edu/∼wangwind/lag.pdf.

Control and Forwarding Plane Interaction in Distributed Routers

Markus Hidell, Peter Sjödin, and Olof Hagsand

KTH – Royal Institute of Technology,
ELECTRUM 229, SE-164 40 Kista, Sweden
{mahidell, psj, olofh}@imit.kth.se

Abstract. The requirements on IP routers continue to increase, both from the control plane and the forwarding plane perspectives. To improve scalability, flexibility, and availability new ways to build future routers need to be investigated. This paper suggests a decentralized, modular system design for routers, based on control elements for functionalities like routing, and forwarding elements for packet processing. Further, we present measurements on the distribution of large routing tables in an experimental platform consisting of one control element and up to 16 forwarding elements.

1 Introduction and Related Work

The growth of the Internet in combination with the demand for new services rapidly increase the requirements imposed on network systems, such as IP routers. The growing traffic volumes require higher performance of the *forwarding plane*, i.e., the router's capacity to process and forward packets. New services often require both more operations to be performed per packet, and that new protocols are introduced in the routers. The latter fact leads to an increased complexity in the *control plane*, i.e., the software controlling the router.

We believe that the monolithic structure of traditional routers is an architectural limitation when it comes to meeting future requirements. Therefore we take the approach to investigate architectures that allow network systems to be *composed* from multiple *modules* (or *elements*), which communicate through open, well-defined interfaces. Our hypothesis is that such architectures can significantly improve scalability, flexibility, and availability of routers. However, there is a certain cost associated with the decentralized structure, since the internal communication incurs overhead. In this paper we investigate different aspects of the internal communication.

A considerable amount of work has been done on modularization and programmability of network systems in the context of active and programmable networks [2], [3], with the purpose to dynamically modify the packet processing path. Exploring decentralized architectures is in line with both industry and research efforts to improve the scaling of Internet routers. Recent commercial high-performance routers are based on distributed multi-chassis solutions, [6], [7], and the 100 Tb/s router project at Stanford University [5] targets a distributed architecture where line card chassis are connected to an optical switch fabric.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1339 – 1342, 2005.

Within the IETF, the ForCES (Forwarding and Control Element Separation) working group [4] aims at defining a protocol for communication between control functions in a router and packet forwarding functions [10]. Goutadier [11] has evaluated an early proposal of the ForCES protocol, called Netlink2.

Finally, Feamster et al. suggest an approach to separation between routing and forwarding [9]. They propose that a *Routing Control Platform* (RCP) should be used to separate interdomain routing from the individual routers, which mainly should be concerned with route lookups and forwarding of packets.

2 System Design and Implementation

We consider a distributed router consisting of different functional elements. Using the terminology of ForCES [4], the distributed router consists of Control Elements (CEs) and Forwarding Elements (FEs). CEs implement functions such as routing protocols, while FEs perform for example packet forwarding. CEs and FEs are connected to an internal network, which can be built in many different ways, e.g., using high-speed optics or high performance switches. In principle, the internal network could even be a router-based IP-network!

Internal communication protocols coordinate activities between the different elements. We have taken the approach to use the protocols emerging within the IETF as a starting point, and added extensions for our specific purposes. We call the result *Forz* – a protocol with three main parts: association, configuration, and data transfer.

The association part is related to internal element discovery and system configuration. The purpose of the configuration part is to convey configuration commands and event notifications between CEs and FEs. For example, if the routing protocols in the CE compute a new best route for some destination, then the FEs need to be informed of this so that they can change their forwarding tables. Forz configuration is based on distributing Netlink [8] messages over the internal control network. Netlink is an API of for Unix networking applications. The purpose of the data transfer part is to switch data packets between FEs across the internal network.

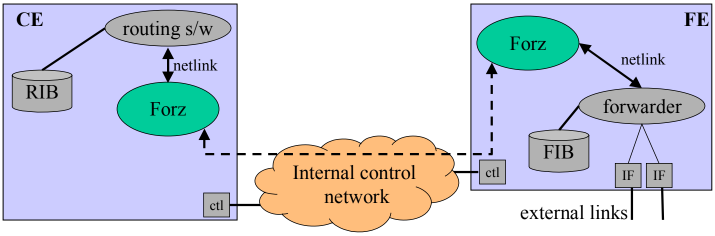


Fig. 1. Physical separation between control element and forwarding element

We have implemented a prototype version of a distributed router consisting of regular PCs connected together by two switched Ethernets (one for control and one for data). So far we have implemented one version of a CE, based on a UNIX system running Zebra open source routing software [**1**], and one FE implementation, also based on PCs running UNIX. The CE is implemented by extending Zebra with the

Forz protocol, and supports the manual configuration of remote network interfaces and static routes. For example, the CE takes commands, given through Zebra's command line interface or a configuration file, and translates them to Netlink messages that are distributed through the Forz protocol (see **Fig. 1**).

3 Performance Evaluation

The purpose of our performance evaluation is to investigate how different transport mechanisms affect the cost in terms of time spent on internal communication. We study a communication-intensive operation, where a large routing table of 100K entries is distributed from the CE to the FEs. We only measure the communication time between the CE and the FEs. Table updates in the FEs are not included.

Forz can run on top of UDP as well as TCP. The choice of transport protocol depends on the type of information that is communicated over the internal network. The distribution of a routing table means that information is duplicated from a sender (CE) to many receivers (FEs). Thus, it appears to be an ideal candidate for multicast transmission. To investigate this, we compare UDP multicast to two unicast mechanisms: UDP unicast, and TCP (which is always unicast).

In contrast to TCP, UDP lacks mechanisms for flow control, congestion control, and segmentation. So in order to use UDP, we have to add support for those mechanisms in Forz. For example, we quickly discovered that if we use UDP without any flow control, a large portion of the route updates are lost since the receivers (the FEs) cannot process the incoming messages fast enough. Furthermore, we also found that the packet size has significant influence on UDP performance. However, congestion control has not been needed in our measurements, since they have been performed in a controlled environment without congestion.

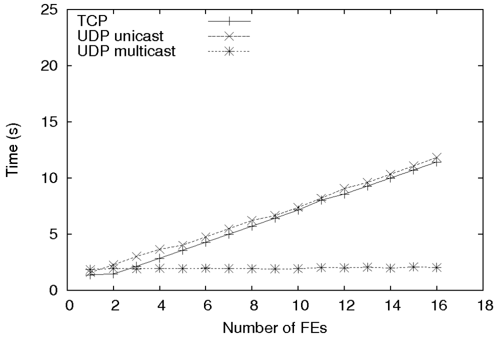


Fig. 2. Total routing table distribution time for different transport mechanisms

We study how the routing table distribution time varies with the number of FEs, and the results are shown in Fig. 2. We observe that, for unicast transport, there is a linear increase in the time it takes to distribute the routing table to all FEs when the number of FEs increases. For UDP multicast, the time is constant, independently of the number of FEs. The value is roughly the same as for UDP unicast with one FE.

Thus, in this experimental set-up, the system can distribute roughly 50,000 route updates per second when UDP multicast is used as the transport mechanism.

For comparison, we also measure TCP. Thus, the transport is reliable and there is no need for Forz level ACKs and segmentation. The drawback is that multicast transfers cannot be used. The results show that the total time to distribute the routing table is slightly lower for TCP than for UDP unicast, which can be explained by TCP's more efficient flow control mechanism and more optimal segmentation.

4 Conclusions and Further Work

We have examined how the time for internal communication depends on the number of FEs. From a scaling perspective, the ideal result is an internal communication time that is independent of the number of FEs, as in the case with UDP multicast. Further, mechanisms such as flow control and segmentation are crucial for the performance. For the unicast-based transport mechanisms, the total distribution time increases linearly with the number of FEs. Such an increase is far from ideal, but it is predictable and may in some cases be manageable even for large systems. The implementation of flow control and segmentation is specific to the configuration we have measured. A more general solution would be to use a reliable multicast protocol. Evaluating such protocols for transport of Forz messages is ongoing work.

References

1. GNU Zebra, URL=http://www.zebra.org
2. Computer Networks Special Issue on Programmable Networks, Vol. 38, No. 3, Feb. 2002.
3. IEEE Journal on Selected Areas in Communications on Active and Programmable Networks, Vol. 19, No. 3, Mar. 2001.
4. ForCES (Forwarding and Control Element Separation) IETF Working group, URL=http://www.ietf.org/html.charters/forces-charter.html.
5. I. Keslassy, et al, "Scaling Internet Routers Using Optics", ACM Sigcomm 2003, Karlsruhe, Germany, 2003.
6. Juniper, "T-Series Routing Platforms: System and Packet Forwarding Architecture", White paper, 2002.
7. Cisco, "Next Generation Networks and the Cisco Carrier Routing System", White paper, 2004.
8. J. Salim, A Kleen, and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", Internet RFC 3549, Jul. 2003.
9. N. Feamster, et al, "The Case for Separating Routing from Routers", ACM SIGCOMM FDNA Workshop, Portland, Oregon, USA, Aug. 30, 2004.
10. A. Doria et al, "ForCES Protocol Specification", IETF Internet Draft draft-ietf-forces-protocol-01.txt, Oct. 2004.
11. G. Goutaudier, "Enhancements and Prototype Implementation of the ForCES Netlink2 Protocol", IBM Research Report RZ 3482 (# 99522), Sep. 2003.

Performance Analysis of 802.11 WLANs Under Sporadic Traffic*

M. Garetto and C.-F. Chiasserini

Dipartimento di Elettronica, Politecnico di Torino, Italy

Abstract. We analyze the performance of 802.11 WLANs that employ the Distributed Coordination Function (DCF). We consider contending stations within radio proximity, and investigate the case in which stations operate under non-saturated conditions. Our modelling technique can be used to study several important issues in 802.11 networks, such as the impact of bursty traffic and the system performance in a multirate environment. The accuracy of the analytical results is verified by simulation with *ns-2*.

1 Introduction

We present an analytical model of the 802.11 DCF that differs from previous work [1–6] in addressing all of the following issues: *(i)* it identifies the critical assumptions in the development of analytical models of 802.11 networks, and presents a fairly simple as well as accurate model of the DCF in presence of non-saturated traffic sources; *(ii)* it is general enough to account for different arrival processes and traffic patterns, in particular, it applies to the case of bursty traffic like that produced by the TCP protocol; *(iii)* it evaluates the system performance in a multirate environment; *(iv)* it applies to the case where a station seizing the channel is entitled to transmit a burst of packets, as specified in the IEEE 802.11e draft standard; *(v)* it evaluates several metrics of interest, such as the network throughput, the packet loss probability, the distribution of the MAC queue length at the wireless stations, and the average packet delay.

2 Modelling 802.11 WLANs

First, we briefly present the basic model describing the behavior of saturated sources; then, we describe two different approaches to deal with the more complicated case in which the stations transmission queue may become empty. Please refer to [7] for further details.

2.1 Modeling Saturated Sources

We follow the approach of [1]; in particular, we assume that the channel has three possible states: *(i)* busy channel due to a successful transmission; *(ii)* busy channel due to a

* This work was supported through the PATTERN project.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1343–1347, 2005.

collision; *(iii)* idle channel. We build a discrete Markov chain, embedded in the temporal evolution of the channel at the instants of a possible state change. By relying on the fundamental assumption that the state of a station is independent of that of the others, we can reduce to considering the behavior of a single, tagged station. Thus, we propose the simple Markov chain depicted in Figure 1. States labeled with b represent the station with backoff counter equal to 0, i.e., the case where the station actually transmits a frame in the current step. States labeled with B model the station while it decrements its backoff counter. States have an index in the range $\{\,0\ldots m\,\}$ representing the "backoff stage", where m is the *maximum retry limit* for the frame to be transmitted. We denote the stationary distribution of the Markov chain by $\boldsymbol{\pi} = \{\pi_s\}$, where s is a generic state of the model. The main differences between our representation and previous models for saturated sources are as follows: *(i)* In case of collision, a station with probability $\alpha_i = 2/W_i$ immediately retransmits the frame in the following step, being W_i the value of the updated contention window; *(ii)* All states belonging to backoff stage i and having backoff counter greater than one, have been collapsed in a single state B_i so as to reduce the number of states. The side-effect of this simplification is that the number of steps waited while decrementing the backoff counter is modeled as a geometrically distributed random variable, instead of a uniformly distributed variable. However, the transition probabilities $P(B_i, b_i)$'s have been chosen in such a way that the stationary probability of the collapsed states B_i are exactly the same that would result considering a uniformly distributed backoff, as done in [1]. By doing so, all performance metrics derived with our simplified model coincide with those obtained with a more precise model of the backoff counter based on a uniform distribution. Thanks to the particular structure of the Markov chain, the derivation of the stationary probabilities is straightforward. By following [1], we can derive: (i) the probability τ that a station transmits during a time step; (ii) the *conditional collision probability* p; (iii) the channel state probabilities Π_s, Π_c and Π_σ, i.e., the probabilities that a generic discrete time step is occupied by a successful transmission, a collision, or an idle slot, respectively; (iv) the *aggregate packet throughput* T_P.

2.2 Modeling Non-saturated Sources

We assume that the MAC buffer at the wireless stations receive data packets from the upper layers according to some stationary, external arrival process with rate λ packets/s. For the sake of simplicity, we assume a Poisson arrival process at each station. The model can be easily extended to account for various degrees of traffic burstiness using batch arrivals. We denote by $\Lambda = \lambda\, n$ the total packet arrival rate at the stations queues. The MAC buffer at each station is assumed to be of finite size K. Packets that cannot be stored in the buffer are immediately discarded upon arrival.

As a first attempt, we incorporate in the Markov model presented above the information about the number of packets currently stored in the queue of the tagged station. The resulting model, named model "A", comprises the states belonging to the set $\{b_{i,j}\,,\, B_{i,j}\}$. The two indexes $0 \le i \le m$ and $0 \le j \le K$ stand for the backoff stage and the number of packets currently stored in the buffer, respectively. We can derive the average probability τ that a station transmits in an arbitrary time step; then, by assuming that the state of a station is independent of that of the others, we can write the probabilities Π'_s, Π'_c and Π'_σ, that a time step is occupied by a successful transmission, a collision, or an idle slot, respectively, given that the tagged station does not transmit. From the solution of the model, obtained with a fixed point approximation, we obtain many significant performance measures, such as the aggregate throughput T_P, the entire distribution of the number of packets queued at a station, the average MAC queueing delay, the packet loss probability due to buffer overflow, as well as the discard probability due to the maximum retry limit. However, the obtained results (not shown here for the lack of space) show that the model overestimates the aggregate throughput, especially at the knee of the curve. The error that we encounter in the model is essentially due to the assumption that the state of individual stations are independent. Our conclusion is that *an 802.11 network under non-saturated conditions cannot be correctly analyzed relying on the independence assumption among stations*, but it is necessary to model the joint behavior of the stations, and in particular the evolution of the number of backlogged stations.

Therefore, we extend the description of the state of the system by keeping track of the number of stations having non-empty transmission queue. We describe in detail the behavior of a tagged station using the same states as in *mod A*, but adding to each state of the tagged station an indication of the number k of stations (excluding the tagged one) having at least one packet in the queue. The state space of the resulting model, named model "B", is the set $\{b_{i,j,k}\ ,\ B_{i,j,k}\}$, where the newly introduced index k takes values in the range $[0 \cdots n-1]$. The number k of competing stations during a time step may vary due to the following events: i) one or more of the stations having an empty buffer receive new data to send, thus increasing k; ii) one of the stations having non-empty buffer successfully transmits a packet leaving an empty queue, thus decreasing k by one. Notice that these two events can occur simultaneously during the successful transmission of a packet.

The number of stations that join the competing set during a time step depends on the duration Δ of the step and on the current value of k. During an interval Δ, a station's queue receives at least one packet with probability $q = 1 - e^{-\lambda\Delta}$. The number of stations that join the set of competing stations is thus distributed according to a binomial distribution with parameters $(q, N-1-k)$.

The possible durations Δ of a time step are related to the probabilities that the channel is occupied by a successful transmission, a collision, or an idle slot. To compute such probabilities, we need to specify the probability $\tau(\tilde{C})$ that one of the stations belonging to the competing set transmits in a given time step. This set may include also the tagged station, thus $0 \leq \tilde{C} \leq n$. In a generic state of the model, we have $\tilde{C} = k$ if $j = 0$, whereas $\tilde{C} = k+1$ if $j > 0$. The probability $\tau(\tilde{C})$, $\tilde{C} \geq 1$, is given by,

$$\tau(\tilde{C}) = \frac{\sum_{i=0}^{m} \sum_{j=1}^{K} \pi_{b_{i,j,\tilde{C}-1}}}{\pi_{b_{0,0,\tilde{C}}} + \pi_{B_{0,0,\tilde{C}}} + \sum_{i=0}^{m} \sum_{j=1}^{K} (\pi_{b_{i,j,\tilde{C}-1}} + \pi_{B_{i,j,\tilde{C}-1}})}$$

If $\tilde{C} = 0$, we obviously have: $\tau(0) = 0$. Note that, differently from the models previously described in the paper, the probability $\tau(\tilde{C})$ may vary from one state to another, as it depends on $\tilde{C}$. The tagged station transmits a packet in all states $b_{i,j,k}$, provided that $j \geq 1$. The *conditional collision probability* is given by: $p(\tilde{C}) = 1 - [1 - \tau(\tilde{C})]^{\tilde{C}-1}$; with probability $1 - p(\tilde{C})$, the transmission is successful. In states $b_{i,0,k}$ and $B_{i,j,k}$, the station does not transmit, and the probabilities $\Pi'_s(\tilde{C},k)$, $\Pi'_c(\tilde{C},k)$ and $\Pi'_\sigma(\tilde{C},k)$ that the channel is occupied by a successful transmission, a collision, or an idle slot, respectively, are functions of k and of the total number of competing stations $\tilde{C}$.

The additional parameter that we need to specify is the probability P_E that, upon successful transmission of a packet, a station other than the tagged one finds itself with an empty buffer, thus decreasing the number of competing stations. This quantity turns out to be the most critical to estimate by our model, as we do not maintain state information about the buffer occupancy at each station. An estimate of P_E can be derived by considering this probability to be dependent on both the number $\tilde{C}$ of competing stations and the backoff stage i. In particular, a good approximation is obtained assuming that, whenever the tagged station is at backoff stage i, the other competing stations are at a backoff stage h that differs from i at most by one, i.e., $|h-i| \leq 1$. Under this approximation, we obtain the following estimate of P_E:

$$P_E(\tilde{C},i) = \frac{\sum_{h:|h-i|\leq 1} \pi_{b_{h,1,\tilde{C}-1}}}{\sum_{h:|h-i|\leq 1} \sum_{j=1}^{K} \pi_{b_{h,j,\tilde{C}-1}}} e^{-\lambda T_S} \qquad (1)$$

Figure 2 shows the good agreement among the distributions of the number C of competing stations obtained with model "B" (referred to as *mod B*) and *ns-2*, for various values of Λ, thus proving the accuracy of our approach. The average buffer occupancy (and thus the average queueing delay) is also very well predicted by *mod B*, as shown in Figure 3.

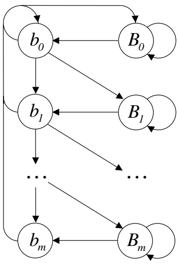


Fig. 1. Markov chain model for saturated sources

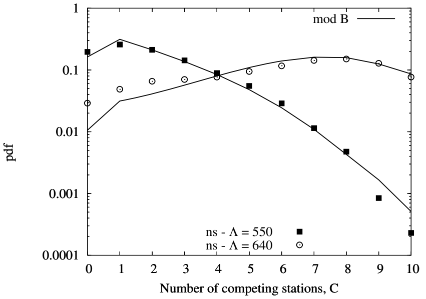


Fig. 2. Distributions of C (competing stations) for $n = 10$ stations, $\Lambda = 550$ and 640 packets/s

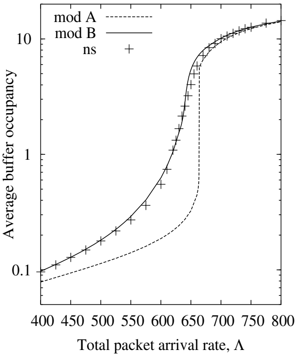


Fig. 3. Average buffer occupancy vs. the total packet arrival rate

References

1. G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE JSAC,* Vol. 18, No. 3, pp. 535–547, Mar. 2000.
2. C. H. Foh and M. Zukerman, "Performance Analysis of the IEEE 802.11 MAC Protocol," *EWC 2002*, Florence, Italy, pp. 184–190, Feb. 2002.
3. E. M. M. Winands, T. J. J. Denteneer, J. A. C. Resing, and R. Rietman, "A Finite-Source Feedback Queueing Network as a Model for the IEEE 802.11 Distributed Coordination Function," *5th EWC*, Barcelona, Spain, pp. 551–557, Feb. 2004.

4. R. Litjens, F. Roijers, J. L. van den Berg, R. J. Boucherie, and M. Fleuren, “Performance Analysis of wireless LANs: an Integrated Packet/Flow Level Approach,” *ITC Conference*, Berlin, Germany, Aug. 2003.
5. O. Tickoo and B. Sikdar, “Queueing Analysis and Delay Mitigation in IEEE 802.11 Random Access MAC based Wireless Networks,” *IEEE INFOCOM*, Hong Kong, Mar. 2004.
6. G. R. Cantieni, Q. Ni, C. Barakat, and T. Turletti, “Performance Analysis of Finite Load Sources in 802.11b Multirate Environments,” *INRIA Research Report RR-4881*.
7. M. Garetto and C.F. Chiasserini, “Performance Analysis of the 802.11 Distributed Coordination Function under Sporadic Traffic,” *Tech. Rep.*, Politecnico di Torino, July 2004.

Virtual Multi-homing: On the Feasibility of Combining Overlay Routing with BGP Routing

Zhi Li, Prasant Mohapatra, and Chen-Nee Chuah

University of California, Davis, CA 95616, USA
{lizhi, prasant}@cs.ucdavis.edu, chuah @ece.ucdavis.edu

Abstract. This paper proposes a framework called *Virtual Multi-Homing (VMH)* to achieve loose source-based path selection and improve inter-domain path diversity. VMH is based on the concept of *Virtual Peering* and *Multi-Homing Overlay* to set up flexible inter-domain relationships. By interacting with BGP whenever possible, VMH can achieve scalable inter-domain route discovery without introducing duplicate work. In addition, VMH is a complementary approach to the existing Internet routing infrastructure and can be incrementally deployed.

1 Introduction

Even though there exists high degree of redundancies in the current Internet topology, Border Gateway Protocol (BGP-4) cannot fully utilize it to provide desirable inter-domain routing services [2]. In some cases, such as content distribution networks (CDNs) and service composition networks, the source Autonomous System (AS) may want to traverse or avoid specific ASes due to performance or security concerns. Unfortunately, the end users and low-tier ASes usually cannot select their preferred inter-domain paths and control how the packets are routed.

To improve path diversity and fault tolerance, some ASes use the concept of multi-homing by subscribing to more than one upstream service providers. However, multi-homing can only increase source ASes' flexibility in selecting the immediate next-hop ASes. In addition, the number of providers an AS can subscribe is usually limited due to economic reasons. Alternative routing architectures have been proposed to improve inter-domain routing performance and provide additional flexibility that BGP lacks, e.g., NIRA [6], RON [2], and RPC [3]. However, these approaches either require the change of the whole Internet routing architecture [6] or are not intended for large-scale deployment [2]. Overlay networks have been proposed as an effective way to improve inter-domain routing performance. However, existing overlays can only reach the destinations within the same overlay networks.

In this paper, we propose a new framework called Virtual Multi-Homing (VMH) to investigate the feasibility of combining the strength of overlay and BGP routing to improve inter-domain routing performance. VMH can be used by overlay service providers, such as Akamai [1] or PlanetLab, to provide wide area

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1348–1352, 2005.

inter-domain routing service to customers. Compared to existing approaches, VMH has the following main advantages. First, VMH enables flexible source-based routing by allowing a source AS to send packets through a series of selected overlay nodes within a multi-homing overlay network (MON). This loose source-based forwarding feature can achieve flexible inter-domain path selection and enhanced QoS support. Second, VMH uses overlay routing technique to enhance inter-domain path diversity for fault tolerance or load balancing. It can also be used to provide proactive re-routing during transient inter-AS path failures without relying on the global path re-convergence. Third, unlike most overlay networks that use a totally separate control plane to discover and exchange path reachability information, VMH interacts with BGP to learn about inter-AS connectivity and routing policies. By associating a subset of the BGP prefixes with a chosen overlay node of MON (e.g., based on proximity to an AS), VMH extends its overlay routing services to a larger range of destinations.

We will introduce the framework of VMH and the preliminary simulation results in Section 2 and discuss the related issues and future work in Section 3.

2 Virtual Multi-Homing (VMH)

2.1 Proposed VMH Framework

In VMH, each AS can have one or more *Multi-Homing Servers (MHS)*. An MHS can either co-exist with a border router or locate at a separate host. An MHS sets up one or more I-BGP sessions with local BGP routers to receive BGP updates and determine its inter-domain paths to other destinations. However, an MHS is a passive peer, i.e., it will not generate route updates to its BGP peers.

The MHSes from different ASes cooperate with each other to form a *Multi-Homing Overlay Network (MON)*. The connections between MHSes are based on the *Overlay Transit* relationships among VMHes. Similar to current inter-AS relationship, *Overlay Transit* determines whether a MHS can forward packets to its neighboring MHSes within the MON. That is, if there is an *Overlay Transit* relationship between two ASes, there is a corresponding overlay link in the MON overlay topology.

A link state based overlay routing protocol runs among the MHSes. This provides resilient overlay routings path connecting each pair of source and destination MHSes. Similar to RON[2], each MHS continuously probes its neighboring MHSes and sends the latest overlay link performance to every other MHS. If there is an IP-layer path failure or service degradation, MON can quickly provide an alternative overlay path.

Virtual Multi-Homing (VMH) runs on top of MON. A new inter-domain transit service called *Virtual Peering* can be set up between two remote ASes. There is a virtual BGP peering session (using an overlay path via MON) between these two ASes' MHSes. One end (an MHS) is a *Virtual Peering Provider (VPP)* while the other end (an MHS) is a *Virtual Peering Customer (VPC)*. Different from *Overlay Transit* relationship, a VPP can send its customer's traffic to any destinations. A VPC can receive BGP updates from its VPPs in addition to the messages for its local BGP router via I-BGP session. However, a VPC will not send any BGP update messages to its VPPs. This restriction ensures that VMH will not introduce any extra routing states into the BGP routing system and affect its performance. Based on MON, an AS can subscribe several VPPs in addition to its directly-connected provider(s). If necessary, it can also subscribe new VPPs on-demand. To send a packet to its destination, a source AS can either (a) send a packet via its direct physical providers, or, (b) send the packet using an overlay path to one of its VPPs, which will then forward the packet to its destination. We can see that the traditional multi-homing service can be deemed as a special case of VMH. It can be observed that VMH can help a source AS to explore inter-domain path diversity, which will potentially improve the inter-domain paths service quality.

Once a VMH user's packet arrives at a MHS, the MHS first picks a destination MHS from its set of VPPs based on its BGP path information. The selection of a destination MHS is based on the distances between VPPs and the packet destination as well as the path disjointness between the default BGP path and the paths via VPPs. Next, the source MHS determines the Loose Forwarding Path (LFP) from itself to the destination MHS using one of the two mechanisms: 1) By default, the source MHS can find an overlay path (list of MHSes) with best service quality (least loss rate or shortest delay); Or 2) the source MHS can find an alternate overlay path through MON based on the constraints specified by packet sender or local routing policy.

After finding the LFP to a destination MHS, the source MHS can encapsulate the original data packets with an additional header which includes the list of MHSes (LFP). When a MHS receives a loose forwarding packet from one of its neighboring MHSes , if the current MHS is the destination MHS on the LFP and the packet source is one of its VPCs, it just removes the LFP header and sends the packet to its destination via the normal IP path. Otherwise, if the packet comes from one of its *Overlay Transit* customers, it locates the LFP from the packet header part and sends out the packet to the next hop MHS. Step-by-step, the packet will be transmitted to the destination MHS following the LFP. It should be noted that, when the data pass through the overlay links, they will be transmitted via IP-Tunneling through corresponding IP-layer paths. However, the destination MHS will remove the LFP packet headers and directly send the data to its destination.

In summary, under VMH framework, the end-to-end routing via VMH is a combination of IP and overlay routing. This will result in less redundant work and is inherently "topology-aware". As shown in [2], the inter-domain forwarding performance can be greatly improved via one or two intermediate forwarding points, we expect that most of LFPs will pass through no more than 3 MHSes. As a result, the extra LFP header will not incur too much overhead.

2.2 Preliminary Performance Evaluation

Our simulation is based on a real Internet inter-domain topology provided by [5] with Tier 5 ASes pruned out, which left a topology with 2473 ASes. For each simulation, we randomly construct a MON whose size varies from 25 to 200 and choose a VPC from those Tier 4 ASes with only one physical provider. We also vary the number of the customer's VPPs from one to eight. For each setup, we use the heuristic method proposed in [4] to choose virtual peering providers. We evaluate VMH's performance in terms of exploring Inter-domain path diversity by the following two metrics [4]:

- End-to-End Path Availability (*EEPA*): it is defined as the ratio of existing a valid inter-domain path from a VPC to a destination given a link/AS availability ratio.
- Resilient Serviced Destination Ratio (*RSDR*): it is the ratio of destinations to which a VPC's paths can benefit from VMH.

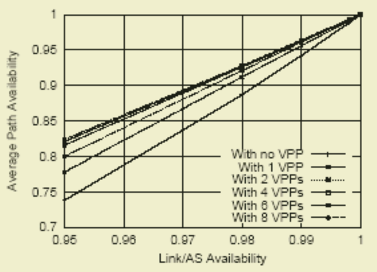


Fig. 1. Path Availability (MON size 100)

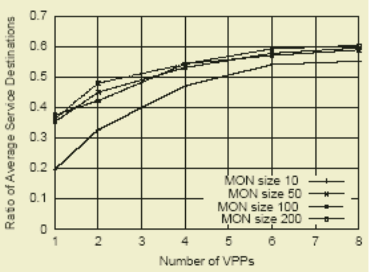


Fig. 2. Average Serviced Dest Paths

Figure 1 shows the average EEPA for a customer AS passing through a *MON* of 100 nodes. For a customer AS, without any VPP, the customer only has one BGP path provided by its service provider. We show that VMH can increase EEPA by providing the customer more flexibility to select loose forwarding path via MON. The EEPA improves with the increasing number of VPPs but saturates when the number of VPPs is more than 4. This is because the VMH's performance is also restricted by the underlying physical topology and the composition of MON, which together determine the maximum benefit VMH can provide to the customer ASes.

Similarly, VMH cannot provide a customer with resilient source-based routing paths (alternate AS-level paths) to all the destinations restricted by the underlying physical topology. Figure 2 shows the average RSDR for a VMH customer. From the figure, we can observe that both MON size and the number of VPPs determine the number of serviced destination paths. This means that careful selection of VPPs is important for a VMH customer.

3 Discussion and Future Work

VMH can effectively utilize the inter-domain path diversity and provide flexible routing service to a large user population. The proposed method utilizes BGP routing states whenever possible, hence reducing the possibility of introducing duplicate work at overlay layer. However, one may raise the following concerns of VMH:

Security Issue from Source-based Routing: In VMH, a *Virtual Peering* session or *Overlay Transit* is set up between two trust-worthy MHSes. An MHS will only deliver packets for its *Virtual Peering* or *Overlay Transit* customers. This can facilitate source tracing and prevent DoS attack without introducing extra security concerns.

Multi-Hop E-BGP Session Stability Analysis over MON: In our technical report [4], we have analyzed the real multi-hop EBGP session stability and shown that overlay can effectively reduce the possibility of session resets. It is shown that the virtual peering BGP session is feasible via MON.

Impact on BGP Routing Performance: We can see that VMH does not introduce any BGP routing messages to the BGP routing system. However, similar to other overlay networks, VMH will certainly affect the intra-domain and inter-domain traffic distribution. How ISPs or BGP can effectively deal with the dynamic traffic shifting is still an open issue.

In future, we plan to deploy a test-bed on top of Planet-Lab to study VMH's performance and related management issues.

References

1. Akamai Corporation. http://www.akamai.com.
2. D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay network. In *18th ACM SOSP*, Oct. 2001.
3. N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. Merwe. The case for seperating routing from routers. In *ACM SIGCOMM FDNA workshp'04*, Sep. 2004.
4. Z. Li, P. Mohapatra, and C.-N. Chuah. Virtual Multi-Homing: on the Feasibility of Combining Overlay Routing with BGP Routing, UC Davis ECS Technical Report, March 2005.
5. L.Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. *Proc. IEEE INFOCOM*, 2002.
6. X. Yang. NIRA: A New Internet Routing Architecture. In *ACM Sigcomm FDNA workshop*, 2003.

Measurement of Rain Attenuation of Microwaves at 12.25GHz in Korea

Dong You Choi

Research Institute of Energy Resources Technology, Chosun University
dy_choi@hanmail.net

Abstract. We present here the results of the measurements of rain-induced attenuation in the vertically polarized signal propagating at 12.25GHz during some rain events, which occurred in the rainy season of the year 2001 at Yong-in, Korea. The attenuation measured experimentally was compared with that obtained using the International Telecommunication Union Radio Communication Sector (ITU-R) model, the SAM model and the Global model. Our measured results are in good agreement with the ITU-R prediction.

1 Introduction

Rain attenuation degrades the performance of the microwave communication system and restricts the use of microwave frequencies for line of sight and space-to-earth communication link. Although fog, clouds and dust particles effect the propagation of signals significantly but rain is the major factor that adversely effects the propagation of signals at frequencies above 10GHz.[1] The knowledge of rain attenuation statistics for the frequency of operation at a particular location is very useful for the planning and engineering of a reliable satellite communication systems.[2]

Thus this paper selected Koreasat-3(116^{o}E), which uses the Ku-band (14GHz/12GHz) frequency, and analyzed the beacon signal level data according to the rain rate from June to August in 2001. Then to provide analysis, the experimental data was compared with that obtained using typical rain attenuation models, such as the ITU-R model[3,4], the SAM model[5] and the Global model[6], which have previously been used in satellite communication systems.

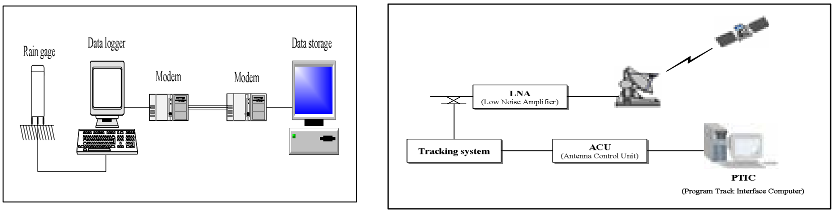
2 Experimental Setup and Measurement Procedure

To measure the rain rate, a rain measurement system was used, which was installed when the Yong-in Satellite Control Office was established. The experimental site in Yong-in, Korea is at 37^{o}43'N and about 142m above sea level. To measure the beacon signal level which is always received at a certain level from a satellite, the controlling equipment of Koreasat-3 was used. Block diagrams for the two measurement systems are shown Fig. 1.

As shown in Fig. 1 (a), the accumulated rain rate data is first collected in a data collector and then is saved in a computer. The rain rate data may be saved in either 10 minute or 10 sec intervals. Since the 10 minute interval data collection was not

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1353 – 1356, 2005.

sufficiently accurate for use with the satellite beacon signal level in accuracy, the 10 sec interval data collection was used. The beacon signal, which is received at a certain level from a satellite, saves the received signal level at intervals of 1 minute, as shown in Fig. 1 (b).



(a) (b)

Fig. 1. Experimental system for measuring (a) the rain rate (b) the beacon signal level

The beacon signal receiving antenna used cassegrain antenna designed specifically for Koreasat-3. In order to compensate for the changes in power level caused by the perturbation of the satellite transmission, which is located in a geostationary orbit, a tracking system using the steptrack tracking method was used.

Table 1. Specifications of antenna

Parameter	Characteristic
Diameter	7.2m
Frequency band	14.0 ~ 14.8GHz(Transmit). 11.7 ~ 12.75GHz(Receive)
G/T at 20° EL	33.5dB/K@11.7GHz
Polarization	Dual Linear Polarized(Transmit, Receive)
Gain	>58.4dBi@14.25GHz
Receive gain	>56.2dBi@11.7GHz
VSWR	1.25:1 Maximum(Transmit, Receive)
Cross poll isolation	>35dB over 1dB BW
Sidelobes	
First sidelobe	<-14dB
Envelope	5log(Degree)dBi for 1<(Degree)<20
1 ~ 20°	35dBi for 20<(Degree)<26.3

Attenuation due to rain over the path was estimated by measuring the excess attenuation over clear weather attenuation values at various rain rates recorded using a tipping bucket rain gauge(diameter 200mm, resolution 0.5mm), which usually provides a good approximation to the instantaneous rain rates. The attenuation of radio wave propagation in a volume of rain of extent L in the direction of wave propagation can be expressed as [7]

$$A = \alpha L \quad [\text{dB}] \tag{1}$$

where α is the specific attenuation of the rain volume, expressed in decibels per kilometer. Specific attenuation can be calculated using raindrop size distribution. The relation of specific attenuation α and rain rate R, is given by Olsen et al. [8] as

$$\alpha = aR^b \quad \text{[dB/km]} \tag{2}$$

The values of the coefficient a and b depend upon frequency, rain temperature and drop size distribution.

3 Results

3.1 Received Beacon Signal Level and Rain Rate

Data corresponding to the signal levels and rain rates during the propagation and communication times were collected and analyzed during the period from June to August, 2001. Rain attenuation is obtained by subtracting a reference level from the measurement of the received beacon signal level. The reference level is obtained by averaging the beacon signal level data obtained during a period in which there is no rain. The 1st to 8th set of statistical data represent the mean values of the data collected on rainy days at intervals of 3～4 days after excluding the minimum and maximum values. The total statistical data was obtained by summing all of the mean data values during each measurement period. Fig. 2 shows the rain attenuation according to rain rate, which was both, measured and calculated using the rain attenuation prediction model.

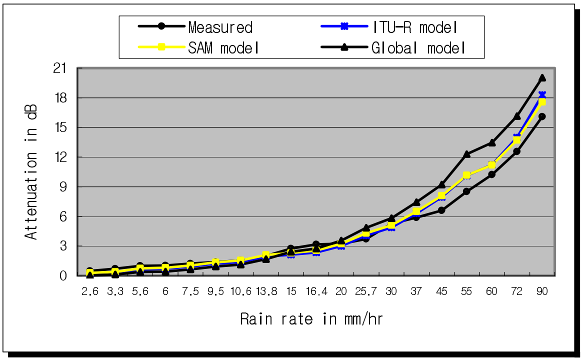


Fig. 2. Comparison of calculated attenuation and measured attenuation

3.2 Discussion

It can be seen in Fig. 2 above, that the experimentally measured attenuation at 12.25GHz is in agreement with that calculated using the ITU-R model, the SAM model, and the Global model for the measured rain rates.

The results shown in Fig. 2 show that, the ITU-R model was in good agreement with the measured attenuation up to a rain rate of about 20mm/hr. When the rain rate exceeds 20mm/hr, the ITU-R model was in excellent agreement with the measured attenuation. As shown in Fig. 2, the SAM model was in excellent agreement with the measured attenuation up to about 20mm/hr, however above this value there is a difference in the attenuation value of above 0.17~1.5dB up to the measured rain rate of 60mm/hr. The Global model shown in Fig. 2 underestimates the attenuation up to a rain rate of about 20mm/hr and, above that rate, it overestimates the attenuation as compared with our experimental results.

4 Conclusions

The rain-induced attenuation at 12.25GHz was measured experimentally. It was found that the experimental results were in good agreement with those obtained using the ITU-R model. At low rain rate, the experimental results were also in close agreement with the SAM model. However, the Global model either underestimates or overestimates the experimentally measured attenuation at the same rain rate. From the above observations, it can be concluded that the rain-induced attenuation at 12.25GHz can be satisfactorily predicted using the ITU-R model for this latitude. However, the rain-induced attenuation statistics cannot be predicted satisfactorily using the SAM model or the Global model for the present location.

References

1. T. C. Ramadorai: Rain attenuation and prediction in the Satellite-Earth Path. in Proc. Workshop HF VHF and Microwave Communications, New Delhi, India(Feb. 1987)
2. Ashok Kumar and I. S. Hudiara: Measurement of Rain-Induced Attenuations of Microwaves at 19.4GHz. IEEE Transactions on Communications, Vol. 1, (JAN., 2002.) 84-86
3. ITU-R: Propagation data and prediction methods required for the design of Earth-space Telecommunications systems. Rec. P.618, ITU-R(1997)
4. ITU-R: Specific attenuation model for rain for use in prediction methods. Rec. P.838, ITU-R(1999)
5. W. L. Stutzman and W. K. Dishman: A simple model for the estimation of rain-induced attenuation along earth-space paths at millimeter wavelength. Radio Science, Vol. 17, no. 6(Nov.-Dec., 1982) 1465-1476
6. R. K. Crane: Prediction of attenuation by rain. IEEE Transactions on Communications, Vol. COM-28, no.9(Sept, 1980) 1717-1733
7. CCIR: Propagation data and prediction methods required for earth-space telecommunication systems. in proc. Plenary Assembly, Vol. 5, Geneva, Switzerland, Rep. 564-4(1990) 447-505
8. R. L. Olsen, D. V. Rogers, and D. B. Hodge: The aR^{b} relation in the calculation of rain attenuation. IEEE Trans. Antennas propagation, Vol. AP-26(1978) 318-329

Scalable Route Selection for IPv6 Multihomed Sites

Cédric de Launois*, Steve Uhlig**, and Olivier Bonaventure

Department of Computing Science and Engineering,
Université catholique de Louvain (UCL), Belgium
{delaunois, suh, bonaventure}@info.ucl.ac.be

Abstract. We propose the use of an improved Internet coordinate system to allow multihomed IPv6 sites to select the source and destination address pair that provides a low delay path. We describe the new coordinate system and evaluate its use on the RIPE test box data set.

1 Introduction

More and more ISPs and corporate networks are multihomed for reliability and performance reasons. Nowadays, at least 60% of stub domains are multihomed. Many sites are expected to require to be multihomed in IPv6. In order to preserve the size of the BGP routing tables in the Internet, every IPv6 multihoming solution is required to allow route aggregation at the provider level. Most IPv6 multihoming mechanisms proposed at the IETF rely on the use of several IPv6 provider-aggregatable prefixes per site [1]. [2] has shown that the use of multiple provider-aggregatable prefixes increases the number of paths available between multihomed sites, such that the number of paths available between two stub ASes is often higher than or equal to 4. The end-to-end delay of a path is a major component of its performances as it reflects its length and bandwidth. Also, TCP performances increase when the delay decreases. Thus, choosing low delay paths is important for most applications, not only for interactive real-time applications.

We propose to use a network coordinate system in order to select, for each source and destination pair, the IPv6 addresses that will lead to a low delay path, without actively probing them. Our goal is to avoid all paths with really bad delays, and to use the lowest delay path as much as possible.

2 The use of Network Coordinate Systems to Identify Paths with Lower Delays

Synthetic coordinate systems have been originally developed to allow an Internet host to predict the round-trip delays to other hosts, without having to

* Supported by a grant from FRIA (Fonds pour la Formation à la recherche dans l'Industrie et dans l'Agriculture, Belgium). This work is also partially supported by the Walloon Government within the WIST TOTEM project.

** Supported by the FNRS (Fonds National de la Recherche Scientifique, Belgium).

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1357–1361, 2005.

contact them first. Each host computes its synthetic coordinates such that the distance between the coordinates of two hosts predicts the RTT between them. For example, if two hosts have coordinates x and y respectively, the distance $\|x - y\|$ is a good predictor of the RTT between them. IPv6 hosts currently arbitrarily choose between multiple global-scope IPv6 addresses. We propose that hosts in IPv6 multihomed sites select their source and destination IPv6 address based on the delays predicted by synthetic coordinates. The selection of those addresses is made once at the start of each flow (e.g. TCP connection) between two hosts.

A host might compute and publish its own coordinates in the Domain Name System. However, in enterprise or campus networks, hosts whithin a single IPv6 prefix will typically end up with similar coordinates. In such a case, the coordinates can be associated to the prefix itself, as a good estimate of the coordinates of any host within this prefix. When this is not true, the network can be divided into blocks, each with its own coordinates, shared by all the hosts whithin the block. We propose that the DNS server of a site computes the coordinates for the few prefixes assigned to the site, and publishes them in the DNS. The coordinates can be advertised using a new DNS resource record. This resource record can possibly be associated directly with the domain name of a host, so that the coordinates and the domain name of a host can be provided together in a single DNS response message, when the DNS name is resolved.

The best path can thus be predicted using a single DNS request, instead of performing multiple series of delay measurements, each using several probes. Since the coordinates should not change frequently, they can be cached in the DNS resolvers, further reducing the cost associated to the prediction of the best path. Another option is that the DNS server makes the choice on behalf of the host by removing bad addresses from the DNS response message. In this case, no modification is required for the hosts.

In this paper, the coordinates are computed by a modified Vivaldi algorithm, because it is simple and fully decentralized [3]. A detailed description of the solution and of the evaluation is available in [4].

2.1 Computing Stable Synthetic Coordinates

Unfortunately, the Vivaldi algorithm does not produce stable coordinates for the nodes when latencies do not satisfy the triangle inequality [4]. This happens quite often in the Internet, for example due to policy-based routing with BGP. Such behaviour is unacceptable in our case since it would require to constantly update the DNS with new coordinates. We propose two modifications to the original Vivaldi algorithm. The first is to improve the local error estimate, so that each node computes more accurate coordinates. The second is to introduce a *loss* factor to prevent the system from oscillating indefinitely. The new algorithm, called SVivaldi, is presented in Alg. 1 and detailed in [4]. It can be shown that the new local error estimator allows to find better solutions, and that the use of a *loss* factor allows to produce stable coordinates [4].

Alg. 1. The SVivaldi algorithm

```
SVivaldi($rtt_{ij}$, $x_j$, $e_j$)
    $w = e_i/(e_i + e_j)$
    $Neigh_i = Neigh_i \cup \{j\}$
    $Rtt_i = Rtt_i \cup \{rtt_{ij}\}$
    $e_i = \frac{\sum_{j \in Neigh_i} |\|x_i - x_j\| - rtt_{ij}|}{\#Neigh_i} \times c_e + e_i \times (1 - c_e)$
    $loss = c_l + (1 - c_l) * loss$
    $\delta = c_c \times w \times (1 - loss)$
    $x_i = x_i + \delta \times (rtt - \|x_i - x_j\|) \times \frac{(x_i - x_j)}{\|x_i - x_j\|}$
```

3 Evaluation of the Quality of the Route Selection

We evaluate here the quality of the routes that are selected using the coordinates computed by SVivaldi. We use delay measurements provided by the RIPE NCC Test Traffic Measurements Service to compute the coordinates of 58 nodes. IPv6 multihoming with multiple prefixes is not currently deployed in the Internet. In order to simulate IPv6 multihoming, we follow a similar methodology to the one used in [5]. A virtual multihomed site is emulated by using collectively a few RIPE nodes in the same metropolitan area. This method models IPv6 multihoming where the provider-dependent prefixes advertised by the virtual site are aggregated by its providers. 13 multihomed sites are emulated, a number similar to the study of Akella et al. on multihoming [5]. 10 sites are dual-homed, 1 is 3-homed, 1 is 4-homed and a last one has 8 providers.

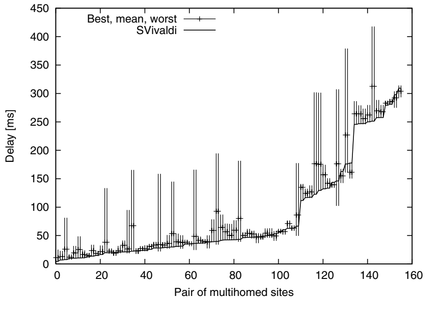


Fig. 1. The delay of the path chosen by SVivaldi for each pair of multihomed sites, in the RIPE data set

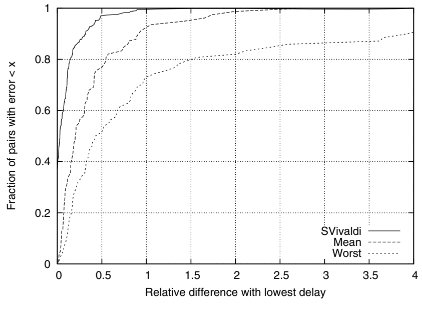


Fig. 2. Differences between the delay of the best path and the delay of the path selected using SVivaldi coordinates

Fig. 1 shows the delay of the path chosen by SVivaldi for each pair of multihomed sites, sorted by increasing delay. The bars indicate the delay of the best, mean and worst path. We can see that the delay of the worst paths can sometimes be several times larger than than the delay of the best path. In this data set, SVivaldi never selects those really bad paths. For the large majority of multihomed pairs, SVivaldi even manages to select almost the best path. When SVivaldi does not select the best paths, the difference between the delay of the path selected and the best delay is not that large. Fig. 2 shows the relative difference between the path with the lowest delay and the path selected by different path selection algorithms. It shows $f(x)$, the fraction of pairs of multihomed sites where a relative difference lower than x is observed. We see that SVivaldi finds the absolute best path in about 40% of the time, and selects a path with a delay at most 20% worse than the best delay for more than 85% of the pairs of multihomed sites. Note that in IDMaps [6], a path selection is considered correct if the delay of the selected path is within a factor of 2 times the delay of the best path. Following this criteria, SVivaldi practically never selects a wrong path. Fig. 2 confirms that SVivaldi succesfully manages to avoid all really bad paths, i.e. paths where the delay is more than twice the best delay. According to Fig. 2, the worst delay is more than twice the best delay for about 25% of IPv6 multihomed sites pairs, so these bad paths are not unusual.

4 Conclusion

With IPv6, the use of multiple prefixes increases the number of paths available to a multihomed site. Selecting the path with the lowest delay is important for many applications. A first contribution is to propose the use of a network coordinate system as an efficient and scalable way to help IPv6 hosts select the best source and destination IPv6 prefixes. We have shown that the use of synthetic coordinates is a scalable way to select good paths and to avoid all really bad paths. Our experiments with the RIPE data set have shown that we can select paths with a delay at most 20% worse than the lowest delay for more than 85% of the pairs of multihomed sites. A second contribution is SVivaldi, an improved version of the Vivaldi algorithm for computing synthetic coordinates. We have introduced two modifications in order to stabilize and produce more accurate coordinates.

Acknowledgments

We thank the RIPE NCC for providing the Test Traffic Measurements Service.

References

1. Huston, G.: Architectural approaches to multi-homing for IPv6. Internet Draft, IETF (2004) <draft-ietf-multi6-architecture-02.txt>, work in progress.
2. de Launois, C., Quoitin, B., Bonaventure, O.: Leveraging Network Performances with IPv6 Multihoming and Multiple Provider-Dependent Aggregatable Prefixes. In: QoS-IP 2005, LNCS 2856, pp.118-179, Catania, Italy (2005)
3. Dabek, F., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proceedings of ACM SIGCOMM'04, Portland, Oregon, USA (2004)

4. de Launois, C., Uhlig, S., Bonaventure, O.: Scalable route selection for IPv6 multihomed sites. http://www.info.ucl.ac.be/people/delaunoi/networking05/ (2005)
5. Akella, A., et al.: A comparison of overlay routing and multihoming route control. In: Proceedings ACM SIGCOMM'04. (2004)
6. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A global internet host distance estimation service. In: Proceedings of IEEE/ACM Transactions on Networking. (2001)

On the Efficient Resource Allocation for High Quality Video Streams and FTP Traffic over Next Generation Wireless Networks

Polychronis Koutsakis and Spyros Psychis

Department of Electronic and Computer Engineering,
Technical University of Crete, 73100 Chania, Greece
{polk, psycho}@telecom.tuc.gr

Abstract. In this paper we propose a Medium Access Control (MAC) protocol for the efficient integration of high quality video traffic with ftp data packet traffic over a wireless channel of very high capacity. Via an extensive simulation study, we evaluate the system's performance under a variety of possible loads which consist of actual MPEG-4 streams and ftp sessions.

1 Introduction

Within a microcell, spatially dispersed mobile terminals (MTs) share a radio channel that connects them to a fixed Base Station (BS). Fixed-length packets arriving at the mobiles are buffered at the terminals until they are transmitted on the uplink to the BS. The BS allocates channel resources, delivers feedback information and transfers the packets to the wired networks and the Internet. In this work, MTs are considered to be high performance devices with extended storage capabilities which can act like cache memories, streaming multimedia material (video, data) to other MTs in the same or in adjacent microcells, in order to reduce client start-up latencies and improve network performance. Therefore, we envision high quality stored video streaming on the uplink channel (MTs to the BS) in order for these streams to be delivered for playback to their destinations.

The delivery of high quality stored video material is a service with very strict QoS requirements. Consequently, the design of a MAC scheme which integrates this class of service with others, often requiring contradictory QoS guarantees, is necessary. In this study we design and evaluate a MAC scheme for the uplink wireless channel, as in the downlink channel the BS is the sole transmitter, and therefore is in complete control of the downstream traffic.

2 Traffic Models

In our study, we use the trace statistics of actual MPEG-4 streams from [1]. The video streams (Silence of the Lambs, The Simpsons, Soccer, Star Wars, Parking Lot Camera) were carefully selected in order to cover a broad range of movie characteristics. New video frames are generated every 40 msecs. We have made the assumption that all packets of a video frame (VF) must be delivered before the next VF arrives. The allowed video packet dropping probability is set to 0.0001 [2].

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1362 – 1365, 2005.

Regarding data traffic, we use the web traffic model presented in [3], the parameters of which result in an average web request size of about 50 KB. As in the reference model of UMTS [4], an FTP file transfer can be described as a web session with only one web request. The arrival process of FTP sessions is Poisson with rate λ sessions per second. An upper bound of one minute is set on the average ftp session delay as a user QoS requirement. This is not a strict requirement, given that the average ftp session size of 50 KB needs in our system a mere 2% average use of the channel information bandwidth per frame, in order to satisfy the specific bound.

3 Frame structure, Base Station Scheduling, Actions of the Mobile Terminals and System Parameters

The uplink channel time is divided into time frames of equal length. Each frame consists of two types of intervals. These are the request intervals and the information intervals. Each of these intervals is divided into a number of time slots.

Within an information interval, each slot accommodates exactly one, fixed length, packet that contains video or ftp data information and a header. Each request slot is subdivided into two mini-slots and each mini-slot accommodates exactly one, fixed length, request packet. By using more than one minislot per request slot, a more efficient usage of the available request bandwidth is possible [2]. The base station broadcasts a short binary feedback packet at the end of each mini-slot indicating only the presence or absence of a collision within the mini-slot (collision (C) versus non-collision (NC)). Upon successfully transmitting a request packet the terminal waits until the end of the corresponding request interval to learn of its reservation slot (or slots). If unsuccessful within the request intervals of the current frame, the terminal attempts again in the request intervals of the next frame.

Video and ftp terminals share the request slots. The two-cell stack reservation random access algorithm [5] is used to resolve the collisions, among video request packets. After the end of the video contention period, the two-cell stack random access algorithm [5] is used to resolve the collisions among ftp terminals with sessions ready for transmission, due to its operational simplicity, stability and high throughput.

When a video terminal wants to decrease its bit rate, it releases all the slots that were previously allocated to it and are no longer necessary. The BS consequently allocates the newly released slots to any other requesting terminals, after the end of the request interval of the next frame. When the bit rate of a video terminal increases, then the terminal must enter the contention process in order to acquire the additional information slots it needs.

The BS allocates channel resources at the end of the corresponding request interval. Video terminals have higher priority in acquiring the slots they demand. In the case that a new VF arrives and the number of ftp slot reservations is such that the video terminal can not be fully serviced, the BS preempts ftp reservations in favor of video terminals waiting for transmission. If a full allocation to the video terminal is still not possible, the BS grants to the video terminals as many of the slots they requested as possible (i.e., the BS makes a partial allocation). The BS allocates the earliest available information slots to the video terminals, which, if needed, keep these slots in the following channel frames, until the next video frame (VF) arrives.

The channel rate is 20 Mbps (from [6]), and the frame duration equal to 12 ms (from [2]). The 12 ms of frame duration accommodate 566 slots (556 information slots plus 10 request slots).

4 Error Model

We use a simplified Fritchman Markov model (from [7]) to emulate the process of packet transmission errors. The Markov model used (presented in Figure 1) comprises of 6 states. State s_0 represents the "good state" and all other states represent the "bad states". When the channel is in state s_0, it can either remain in this state or make the transition to state s_1 (with probability p_0). When the channel is in a bad state, the transition is either to the next higher state or back to state s_0, based on the status of the currently received packet. This means that the channel does not remain in any of the "bad states" for more than 1 slot. With this model, it is only possible to generate burst errors of length equal to 5 slots at most. The transition probabilities (p_0, p_1,...p_5) of the error model are (0.0000446, 0.100324, 0.164083, 0.149606, 0.526316, 0), respectively. The probability that the channel is in a good state is p_{good}=0.99995, and the total probability for a transition from a bad state to the good state is $p_{bad\text{-}good}$ =0.8924.

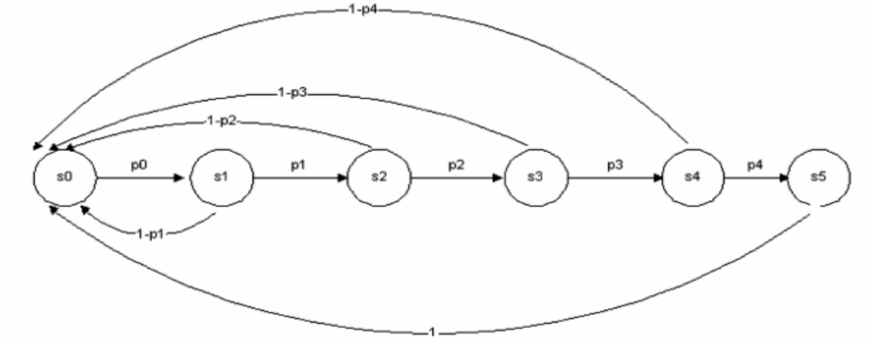


Fig. 1. Error Model

5 Simulation Results and Discussion

Via an extensive simulation study we studied system behavior and performance under all possible movie loads from 1 to 6 movies. For each load, all different combinations were examined (63 in total), from 10 times each (Monte Carlo method).

Table 1 shows the average results of the simulation runs under all possible movie loads. As the number of video terminals increases, the aggregate bit rate gets smoother, since the superposition of the movies is known from the literature to always be less bursty than a single movie, and the average bandwidth of the superposition is close to the sum of the mean bit rates of the movies. This is proven once more in our study, as the case of the system accommodating only one movie is shown to provide the smallest throughput than any other case of movie load. The λ parameter corresponds to the maximum ftp session

Table 1. Integration Results for the superposition of movies

System Load	Average λ (ftp sessions/frame)	Average Channel Throughput (%)
1 Movie	0.312	81.32
2 Movies	0.272	81.99
3 Movies	0.207	83.64
4 Movies	0.158	88.75
5 Movies	0.087	95.28
6 Movies	0.019	93.54

arrival rate (expressed in sessions per frame) sustained by the system (i.e., such that the video packet dropping probability<0.0001 and the average ftp session delay < 1 minute). Our results show that the proposed mechanism achieves very high aggregate channel throughput (steadily more than 80%) for all cases of combined video traffic load, while preserving the Quality of Service (QoS) requirements of both video and ftp traffic types.

The reason that the channel throughput decreases when six videos are active compared to the case of five active videos, can be explained if we sum up the peak bit rates of the six movies; the total peak rate reaches 25 Mbps, which is significantly higher than the total channel capacity. Of course, it is rather rare that all of the 6 video terminals would transmit at their peak bit rates at the same time but still their superposition surpasses for a few video frames the channel capacity. This forces the system to experience, in these video frames, severe video packet dropping, because of the very strict video dropping probability requirements, and therefore leads to the decrease in throughput in comparison with the 5 active video terminals scenario.

References

1. [Online] http://peach.eas.asu.edu/index.html
2. P. Koutsakis, S. Psychis and M. Paterakis, “On the Integration of MPEG-4 Video Streams with Voice and E-mail Data Packet Traffic over Wireless Picocellular Networks”, in Proceedings of the IEEE PIMRC 2001 Conference, San Diego, California, USA.
3. P. Tran-Gia, D. Staehle, K. Leibnitz, “Source traffic modeling of wireless applications”, International Journal of Electronics and Communications,Vol. 55, No. 1, 2001, pp. 27-37.
4. “Universal Mobile Telecommunication System (UMTS); Selection Procedures for the Choice of Radio Transmission Technologies of the UMTS”, Technical Report TR 101 112 v3.2.0, ETSI, April 1998.
5. A. Cleary and M. Paterakis, ″Design and Performance Evaluation of an RRA Scheme for Voice-Data Channel Access in Outdoor Microcellular Wireless Environments″, *ACM/Baltzer MONET Journal*, Vol. 2, No.1, pp. 31-43, 1997.
6. N. Passas, D. Skyrianoglou and L. Merakos, “Traffic Scheduling in Wireless ATM Networks”, in Proceedings of the IEEE ATM'97 Workshop, Lisbon, Portugal, May 1997.
7. C. Y. Hsu, A. Ortega and M. Khansari, “Rate Control for Robust Video Transmission over Burst-Error Wireless Channels”, IEEE Journal on Selected Areas in Communications, Vol. 17, No.5, May 1999.

A Self-organizing Routing Scheme for Random Networks

Thomas Fuhrmann*

System Architecture Group, Universität Karlsruhe (TH),
76128 Karlsruhe, Germany
fuhrmann@ira.uka.de

Abstract. Most routing protocols employ address aggregation to achieve scalability with respect to routing table size. But often, as networks grow in size and complexity, address aggregation fails. Other networks, e.g. sensor-actuator networks or ad-hoc networks, that are characterized by "organic growth" might not at all follow the classical hierarchical structures that are required for aggregation.

In this paper, we present a fully self-organizing routing scheme that is able to efficiently route messages in random networks with randomly assigned node addresses. The protocol combines peer-to-peer techniques with source routing and can be implemented to work with very limited resource demands. With the help of simulations we show that it nevertheless quickly converges into a globally consistent state and achieves a routing stretch of only 1.2 – 1.3 in a network with more than 10^5 randomly assigned nodes.

Keywords: Self-organization, Peer-to-Peer, Ad-Hoc Routing.

1 The Problems of Routing

Routing is the task of picking a sequence of nodes between a packet's source and its destination. Optimally, this sequence should denote a shortest path, but a slight deterioration is tolerable for most applications. Typically, routing is done fully distributedly in a hop by hop manner, i.e. each node decides independently of all other nodes which of its neighbors is closest to the packet's destination and thus best suited to next route the packet. However, in order to apply this mechanism, all the nodes must have – in some sense – a complete knowledge of the network. More precisely, each node must be able to identify the next hop for *all the addresses in the entire network.*

Today, large scale networks achieve scalability by aggregating node addresses in the nodes' routing tables. This requires the network to be so structured that node addresses and network topology match. (Theoreticians call this "labeled routing" [1].) If networks are not thoroughly planned or if the once well crafted

* This work was supported by Deutsche Forschungsgemeinschaft under grant FU448-1.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1366–1369, 2005.

network structure becomes outdated, aggregation fails, and, as a consequence, the lengths of the routing tables explode [5].

Before the advent of large-scale hierarchical networks people were used to source routing bridges that routed frames in small or medium size networks with random node address distribution. The necessity to flood the network with discovery frames, however, prevented this approach from being scalable. In other words: In unstructured networks with random node addresses classial routing mechanisms like Dijkstra and Bellman-Ford do not scale memory-wise (routing table explosion), whereas classical source routing mechanisms do not scale message-wise (flooding).

However, being able to efficiently route in networks that lack a well-crafted structure ("name independent routing") has several advantages [3]. E.g., such networks do not require any coordinated network planning. Network growth and layman modifications cannot spoil anything. Redundancy ("multihoming") is easily achieved since there are no hierarchical structures that one has to comply with. Moreover, being able to randomly assign addresses to nodes facilitates various applications ranging from cryptographic security to content addressing.

2 Scalable Source Routing

In this paper, we present the evaluation of a novel source routing mechanism that is both message and memory efficient. (The full protocol specification is available as a technical report [2]. The source code of our implementation is available upon request.) The core idea of our proposal is the combination of overlay routing techniques with source routing. Thereby, the knowledge about the network structure is distributed over all nodes in the network, reducing the size of the routing table in each node. Yet the knowledge is efficiently accessible by any node.

We implemented the protocol in C++ and simulated random graph networks with up to 128 000 nodes. We studied several different topologies (small-world, Erdösch-Renyi, unit-disk graph) and found that our protocol is applicable in all of them. Due to their great practical relevance we discuss only small-world networks, here. This class of networks comprises e.g. the telephone network and the Internet. Since our protocol has very low memory and compute resource demands (only 4 KB per node for networks of more than 100 000 nodes), our protocol seems to be very well-suited for sensor-networks.

In our simulations we specifically focused on two aspects: (1) How quick and message efficient does the system converge into its globally consistent state? (2) What is the achieved path length of the source routes as compared to shortest paths that could be obtained by classical routing algorithms? — We also studied a third important question: (3) 'How does the node memory size influence the routing performance?', but found that an increase beyond 4 KB (corresponding to 255 nodes in the source route cache) per node had hardly any effect in our simulations. This is an impressive proof for the overwhelming memory efficiency of our protocol!

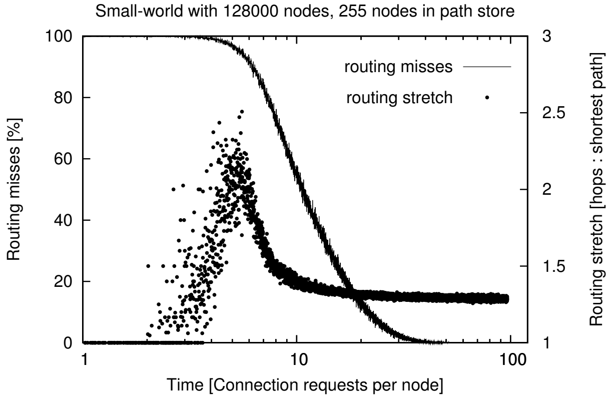


Fig. 1. Convergence and achieved routing stretch

2.1 Global Consistency and Achieved Routing Stretch

Figure 1 shows the convergence of a 128 000 node random graph network where all nodes are initialized at once. (This is a worst case scenario. If nodes are deployed successively, each node almost instantly integrates correctly.) In the simulation, each node issues connection requests for nodes that are drawn randomly (with uniform distribution) from the set of all nodes.

As can be seen, immediately after start-up, the network is inconsistent and almost all connection requests fail. Only the rare requests for addresses in the vicinity of a node can be positively answered. The resulting source routes are typically shortest paths. Hence the low routing stretch at the beginning. After each node has issued about 5–6 connection requests, the network is about to converge. Although now more and more connections can be established, the achieved routing stretch becomes worse and worse peaking at a value of 2.5. After that the knowledge about good paths builds up in the route caches and the routing stretch becomes better and better, finally stabilizing at about 1.2.

2.2 Routing Table Structure and Traffic Concentration

The memory and message efficient properties of routing in distributed hash tables like Chord [4] are based on the fact that the routing table entries of each node point to nodes in exponentially increasing distances. Fig. 2 demonstrates that this distribution is an emergent feature of our protocol. It shows the distribution of address distances between a node and all the the nodes in its route cache (16 000 nodes), binned into 320 bins with exponentially growing size.

Small-world networks contain so-called *hub* nodes that serve as natural landmarks for our protocol and enable the nodes to efficiently cut down the source route paths: If two paths both happen to contain the same two hub nodes, both paths can share their knowledge about a good path between these two hub nodes. One might think that this leads to a traffic concentration at the hubs. Fig. 2

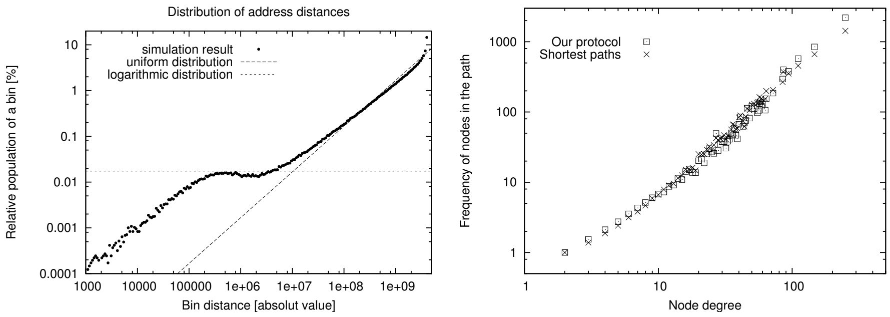


Fig. 2. Distribution of node distances (left) and node frequency in the paths (right)

show that the latter *is not the case*. Our protocol leads to the same distribution that classical shortest path algorithms achieve.

3 Conclusion and Outlook to Future Work

In this paper we presented the evaluation of a novel routing mechanism that breaks the message – memory efficiency trade-off of classical routing mechanisms. With the help of simulations we have demonstrated that the protocol is capable of achieving a routing stretch of only about 1.2 for large random networks containing nodes with random addresses. The ability to efficiently route in such environments makes our protocol especially suitable for networks that are not centrally planned or managed, and that thus do not obey strong hierarchies. Its low resource demands recommend our protocol for sensor-actuator networks where lots of small nodes need to communicate in a self-organizing manner.

Extensions of this protocol that are currently under development will improve the protocol's capability to support node mobility and prevent malicious nodes from harming the overall routing stability.

References

1. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In *Proc. 16th ACM symposium on Parallelism in algorithms and architectures*, Barcelona, Spain, 2004.
2. Curt Cramer, Thomas Fuhrmann, and Kendy Kutzner. Scalable Source Routing Protocol Specification. Technical Report 2005-4, University of Karlsruhe, 2005.
3. Bryan Ford. Unmanaged Internet Protocol. *ACM SIGCOMM Computer Communications Review*, 34(1):93–98, January 2004.
4. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM 2001*, pages 149–160, San Diego, California, 2001.
5. Don Towsley Tian Bu, Lixin Gao. On characterizing BGP routing table growth. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 45(1), May 2004.

A Novel Extension for On-demand Routing Protocol in Event-Driven Sensor Networks[*]

Dong-Hyun Chae, Kyu-Ho Han, Kyung-Soo Lim, and Sun-Shin An

Department of Electronics Engineering, Korea University,
1,5-ka, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
{hsunhwa, garget, angus, sunshin}@dsys.korea.ac.kr

Abstract. We describe a problem of redundant flooding induced by multiple sensor nodes during route discovery in event-driven wireless sensor networks, and propose a novel extension to the on-demand ad hoc routing protocol, in order to reduce the number of signaling messages during the route discovery phase. That is, our extension reduces energy consumption during route discovery. The heuristically and temporarily selected Path Set-up Coordinator (PSC) plays the role of a route request broker that alleviates redundant route request flooding. The simulation results show that our extension not only helps to conserve energy, but also reduces the disruption caused by the broadcast storm.

1 Introduction

In this paper, we describe the broadcast storm problem [1] that is susceptible to arise during the route discovery phase as a result of redundant flooding from another viewpoint. This problem arises when several sensor nodes broadcast route request packets (RREQs) to the same destination (i.e., the sink node) in an event-driven sensor network in which the on-demand ad hoc routing protocol is utilized.

In an event-driven sensor network, several sensor nodes can detect an event simultaneously. The situation will inevitably arise in which multiple source nodes try to find route paths toward the same destination by RREQ flooding simultaneously in such a case. This redundant network-wide flooding causes considerable energy consumption and frequent MAC layer collision in a densely deployed sensor network.

Sensor nodes are energy constrained devices and therefore, it is crucial to minimize such redundant network-wide flooding. In order to overcome this problem, we propose a novel extension to the on-demand ad hoc routing protocol, in order to reduce the number of signaling messages during the route discovery phase.

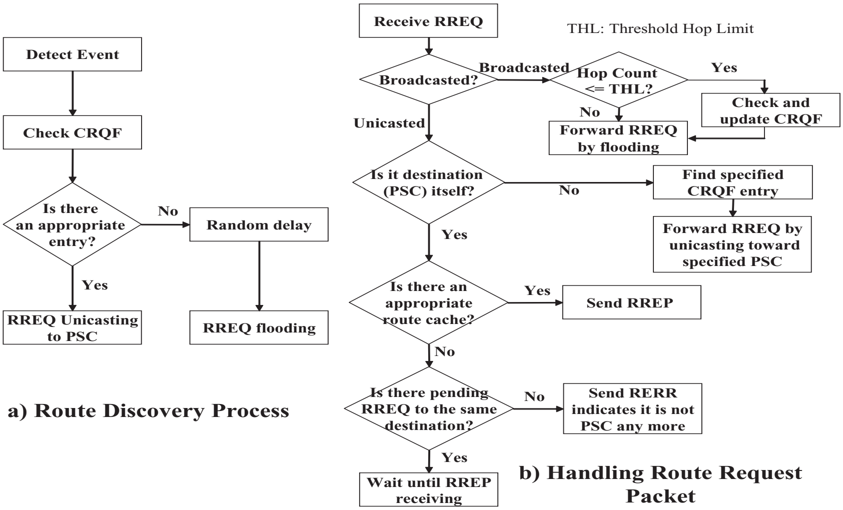
[*] This work was partially supported by CUCN (National Center of Excellence in Ubiquitous Computing and Networking).

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1370–1373, 2005.

2 Our Proposal

The main idea is very simple. If a node needs to obtain a route path to the sink node, it sends RREQs by the unicasting mechanism to the nearest node which has a route cache or which has a pending route discovery that was initiated by network-wide flooding.

Figure 1 shows a flow chart of our proposed algorithm. Figure 1-a) describes route discovery process and b) shows how to dispatch a received RREQ.


Detect Event
Check CRQF
Is there an appropriate entry?
No
Random delay
Yes
RREQ Unicasting to PSC
RREQ flooding
Receive RREQ
THL: Threshold Hop Limit
Broadcasted?
Broadcasted
Hop Count <= THL?
Yes
Check and update CRQF
No
Forward RREQ by flooding
Unicasted
Is it destination (PSC) itself?
No
Find specified CRQF entry
Forward RREQ by unicasting toward specified PSC
Yes
Is there an appropriate route cache?
Yes
Send RREP
No
Is there pending RREQ to the same destination?
No
Send RERR indicates it is not PSC any more
Yes
Wait until RREP receiving

a) Route Discovery Process

b) Handling Route Request Packet

Fig. 1. Flow Chart of the Proposed Extension to the On-Demand Ad Hoc Routing Protocol

The Path Set-up Coordinator (PSC) is defined as the node that sends RREQs by network-wide flooding. All of the other neighbors and nodes which are located within the Threshold Hop Limit (THL) send RREQs by the unicast mechanism to the PSC.

The PSC is selected using a simple formula. It is simply the first node that tries to obtain a route path to the sink node in its locality. The PSC does not have permanent responsibility for replying to unicast RREQs. The PSC continues to perform this role, only as long as it has an active route cache. This heuristic approach is appropriate for the mechanism of Path Set-up Coordinator (PSC) selection. This approach eliminates the necessity to use an additional protocol and signaling message to select a PSC.

In order to be able to perform the unicast route set-up process, a node has to maintain a Cache of the RREQs that it receives by Flooding (CRQF). CRQF would need to contain the source node ID, timestamp, hop count, previous node ID or reverse source routes for source routing protocol, destination node ID (i.e. sink node ID) of the RREQ and current time. This entry can be managed by a soft-state policy and may be updated by another RREQ received by flooding.

3 Performance Evaluation

We evaluated the proposed scheme using the ns-2 simulator [2], in order to evaluate its performance in terms of the energy conservation afforded by the reduction in the number of signaling messages.

We generated 10 topologies in an area of 200m by 200m, in which 150 fixed nodes were randomly located. There was one randomly selected sink node in each topology.

We used the 802.11 MAC layer protocol and reduced the propagation range to 40m. The simulation time was limited to 100 seconds. We forced an event to move toward a randomly selected location. We selected AODV [3] as on-demand routing protocol.

We compared three different modes of the AODV as described below:

1. Default AODV (AODV): This is the default implementation of the AODV in ns-2. It is substantially implemented in Internet RFC 3561.
2. Modified Expanded Ring Search of AODV (MERS): Expanded Ring Search is implemented and utilized in Default AODV. We modified and adjusted this default expanded ring search implementation so that the strategy of the expanded ring search can be applied to the initial path setup phase without the necessity for the old path information.
3. AODV modified according to our suggestion (OURS): This is the extended implementation, incorporating our proposed mechanism. The proposed algorithm was implemented using the THL value that is indicated by K in table 1. As the value of K, that is THL, increases, the total number of the signaling messages decreases. However, if THL were larger, many nodes would not take the optimal route path. Therefore, the appropriate THL value should be selected on the basis of sufficient topology information.

The simulations were performed 10 times for each of the 10 different topologies and tables 1 shows the average values of each of the measured numerical factors, as described below:

1. Total Signaling Messages: This factor indicates the total number of signaling messages that are transmitted and received. (i.e., Total Tx + Total Rx)
2. Path Setup Success Ratio: This is the ratio of the total number of attempts to find paths successful or not to the total number of attempts.
3. Path Setup Latency: This means the path setup latency itself.
4. Path Setup Latency up to Success: This indicates the accumulated path setup latency from the initial path setup attempt until successful path setup is accomplished. If the initial path setup attempt fails, the node retries after a specified timeout.

Table 1 shows our simulation result. Using the proposed mechanism, reductions in the number of signaling messages of approximately 66.7% (K=1), 84.8% (K=2), 86.9% (K=3) and 88.6% (K=4) are obtained in comparison with default AODV and 45.5% (K=1), 75.2% (K=2), 78.5% (K=3) and 81.4% (K=4) packet reductions are observed in comparison to the MERS.

Table 1. Simulation result

	Default AODV	MERS	OURS (K=1)	OURS (K=2)	OURS (K=3)	OURS (K=4)
Total Signaling Messages	31799.9	19428.4	10586.7	4826.8	4172.1	3618.6
Path Setup Success Ratio	0.5471	0.4406	0.7116	0.8683	0.8063	0.7898
Path Setup Latency	0.3307	0.1622	0.1473	0.1283	0.1293	0.2217
Path Setup Latency up to Success	1.5062	0.8230	0.5310	0.3173	0.3244	0.5653

The observed values of the path setup latency and the path setup latency up to success show that the proposed scheme causes the sensor nodes to take route paths with less setup latency. The path setup success ratio indicates that the proposed algorithm provides greater freedom from the broadcast storm problem, while simultaneously reducing the energy consumption.

4 Conclusion

The issue of energy conservation is critical in sensor networks, in which energy resources are limited. In this study, we propose a novel extension to the on-demand routing protocol for event-driven, randomly and densely deployed wireless sensor networks.

Our extension is designed to reduce redundant route request packets in event-driven sensor networks. A Path Set-up Coordinator (PSC) is heuristically and temporally selected on demand, and it is only allowed to perform network-wide flooding in a local area. All the other nodes send route request packets to the PSC by unicasting.

This PSC selection and unicasting mechanism reduces the number of required signaling messages and this signaling reduction in turn lowers the energy consumption.

The simulation results demonstrate that the proposed extension can not only conserve energy, but can also reduce the disruption caused by the broadcast storm problem. The proposed protocol outperforms the other protocols in terms of both energy consumption and the successful route discovery rate, as well as lowering the path setup latency in the route discovery phase.

References

1. Y. Tseng, S. Ni, Y. Chen, and J. Sheu, "The Broadcast Storm Problem in Mobile Ad Hoc Network," Wireless Networks, Vol. 8, 153-167, 2002
2. NS-2 Network Simulator, http://www.isi.edu/nsnam/ns
3. C. Perkins, E. Belding-Royer, S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Internet RFC 3561, July 2003

Equivalent Gibbs Duhem Law in Network Queue

Sandeep Sudeep and Joseph Y.Hui

Sandeep Sudeep, Electrical Engineering, Arizona State University,
Tempe, AZ 85287 USA
J. Hui, Electrical Engineering, Arizona State University,
Tempe, AZ 85287 USA
sandeepsudeep@asu.edu,jhui@asu.edu

Abstract. High speed networks are highly dynamical systems. Large deviation techniques have been employed to characterize their loss probability. Large deviation is also used to calculate shadow prices of quality of service of a scalable network queue with respect to buffer size B, capacity C and arrival rate N.In this paper we prove that these shadow prices are independent of (B,C,N) parameters. Gibbs Duhem law of thermodynamics is described. Equivalent Gibbs Duhem type law in scalable network queue is derived. Application of equivalent network Gibbs Duhem law is discussed.

Index Terms: Quality of Service, Macroscopic parameter, Microscopic Parameter, Gibbs Duhem law.

1 Introduction

Quality of service of a network queue is defined as negative logarithm of its loss probability. Loss probability in high speed network is usually defined in large deviation regime. Hui[1] proves the linear scalability of quality of service(Q) of a network queue with respect to capacity(C), number of connections(N) and buffer size(B).Hui[1] defines microscopic and macroscopic entities in a scalable queue equivalent to microscopic and macroscopic parameters in a thermodynamic system. Buffer B, Capacity C, arrival rate N and quality of service Q are defined as macroscopic properties of queue whereas rate of change of one macroscopic parameters with other is defined as microscopic parameter.

In this paper we prove that microscopic parameters of a scalable network queue are related to each other independent of macroscopic parameters of network queue. We describe Gibbs Duhem law in thermodynamics and present equivalent Gibbs Duhem law in a scalable network queue. Gibbs Duhem law relates microscopic entities of a network queue in differential form.

Section 2 presents a brief introduction to Hui[1]. Section 3 presents relationship between microscopic entities. Section 4 discusses Gibbs Duhem law of thermodynamics and present equivalent law in scalable network queue. Section 5 presents the conclusion and future work.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1374 – 1378, 2005.

2 Theory of Scalability

Quality of service of a network system is a scalable function of the macroscopic parameters [1]. Hui proves the linear scalability of quality of service (Q) of a network queue with respect to capacity(C), arrival rate (N) and buffer size (B) that is, if all the macroscopic parameters of the network queue are scaled by a constant α quality of service gets multiplied by the same constant. We pose this scalability property as a postulate.

$$Q\,(\alpha C, \alpha N, \alpha B)=\alpha Q(C,\,N,\,B)\,. \tag{1}$$

The manifold of (Q, C, N, B) can be described by the marginals, namely taking partial derivatives using two of the four quantities while keeping the remaining two quantities constant. We study the differential form of

$$Q=Q(C,\,N,\,B). \tag{2}$$

Differentiating

$$dQ=\left(\frac{\partial Q}{\partial C}\right)_{B,N} dC+\left(\frac{\partial Q}{\partial B}\right)_{C,N} dB+\left(\frac{\partial Q}{\partial N}\right)_{C,B} dN \tag{3}$$

$$\left(\frac{\partial Q}{\partial C}\right)_{B,N} \equiv \theta\ . \tag{4}$$

$$\left(\frac{\partial Q}{\partial B}\right)_{C,N} \equiv \delta\,. \tag{5}$$

$$-\left(\frac{\partial Q}{\partial N}\right)_{C,B} \equiv \mu\,. \tag{6}$$

Where θ, δ, μ are called microscopic/intensive parameters [1]. Hence

$$dQ=\theta dC+\delta dB-\mu dN\,. \tag{7}$$

Since Q is scalable in B, C, N

$$Q=\theta C+\delta B-\mu N\,. \tag{8}$$

3 Relation Between Microscopic Parameters

Microscopic parameters represent the shadow price of quality of service with respect to macroscopic parameters [1]. They reflect the effect on quality of service in case the buffer size, capacity or number of sources of the system changes.

We propose that for a scalable system it is possible to derive a relation among the intrinsic quantities independent of extensive parameters. As example we take the M/M/1/B and M/M/C/C+B queues to illustrate these relationships.

A. M/M/1/B Queue

Quality of service of M/M/1/B [1] is

$$\mathrm{Q}_{M/M/1/B} = \mathrm{B}\ \log\left(\frac{C}{N}\right). \tag{9}$$

Where B is the buffer size, N the total number of connections and C is server capacity.

$$\theta = \left(\frac{\partial Q}{\partial C}\right)_{N,B} = \frac{B}{C}\ . \tag{10}$$

$$\delta = \left(\frac{\partial Q}{\partial B}\right)_{C,N} = \log\left(\frac{C}{N}\right). \tag{11}$$

$$\mu = \left(-\frac{\partial Q}{\partial N}\right)_{C,B} = \frac{B}{N}\ . \tag{12}$$

From equation (10), (11) and (12)

$$\mu = \theta e^{\delta}. \tag{13}$$

$$\log(\mu) = \log(\theta) + \delta. \tag{14}$$

Equation (14) presents relation between microscopic quantities independent of extrinsic quantities.

B. M/M/C/C+B Queue

Quality of service for M/M/C/C+B queue [1] is

$$\mathrm{Q}_{M/M/C/C+B} = [\mathrm{C}\ (1-\frac{N}{C}) + (\mathrm{C+B})\log\rho\ ] = (\mathrm{C+B})\log\left(\frac{C}{N}\right) + \mathrm{N-C}\ . \tag{15}$$

The M/M/C/C+B queue has B waiting room and C number of servers. Quality of service in case of a M/M/C/C+B queue is

$$\mathrm{Q} = (\mathrm{B+C})\log\left(\frac{C}{N}\right) + \mathrm{N- C}\ . \tag{16}$$

$$\theta = \frac{B}{C} + \log\left(\frac{C}{N}\right). \tag{17}$$

$$\delta = \log(\frac{C}{N}). \tag{18}$$

$$-\mu = 1 - \frac{B + C}{N} . \tag{19}$$

From equations (17), (18) and (19) we have

$$\mu + 1 = e^{\delta}(\theta - \delta + 1) . \tag{20}$$

$$\log(\mu + 1) = \delta + \log(\theta - \delta + 1). \tag{21}$$

Equation (21) presents relation between the three microscopic quantities in M/M/C/C+B queue. Hence we prove that microscopic parameters of a scalable network queue are related to each other independent of individual macroscopic parameters.

4 Gibbs-Duhem Type Law In Networks

In thermodynamics there is a relation among the intensive parameters. For a single component system μ chemical potential is a function of T temperature and P pressure.

The existence of a relationship among the various microscopic parameters is consequence of the homogenous first-order property of the fundamental relation of thermodynamics. For a single component system this property permits the fundamental relation to be written in the form *u=u(s,v)* where *u* is the energy per mole, *s* is the entropy per mole and *v* is the volume per mole; each of the three intensive parameters is then also a function of *s* and *v*. A differential form of the relation among the intensive parameters can be obtained directly from the Euler relation and is known as the Gibbs Duhem relation proved in [2].

$$d\mu = -s\,dT + v\,dP . \tag{22}$$

The variation in chemical potential is not independent of the variation of temperature and pressure, but the variation of any one of them can be computed in the terms of the variations of the other two. Corresponding to the Gibbs Duham relation in thermodynamics we derive a Gibbs Duham type relation in scalable network queues. Microscopic quantities in a scalable system are not all independent. There is a relation among the intensive parameters and for a single component system μ is a function of θ and δ. The existence of a relationship among the microscopic parameters is a consequence of homogenous first-order property of the fundamental relation. The Gibbs-Duhem relation presents relation between microscopic quantities in thermodynamics. We find the Gibbs Duhem type relation in a network queue

$$Q = \theta C + \delta B - \mu N . \tag{23}$$

Differentiating

$$d\,\mathrm{Q}{=}\theta dC+\delta dB-\mu dN\ +\mathrm{C}d\theta\ +\mathrm{B}\ d\delta\ \text{-}\,\mathrm{N}d\mu\,. \tag{24}$$

From equation (7)

$$dQ=\theta dC{+}\delta dB{-}\mu dN\,. \tag{25}$$

Subtracting equation (25) from (24)

$$\mathrm{N}d\mu=\mathrm{C}\ d\theta\ +\mathrm{B}\ d\delta\,. \tag{26}$$

$$d\mu=\mathrm{c}\ d\theta\ +\mathrm{b}\,d\delta\,. \tag{27}$$

Where $\mathrm{b}\equiv\frac{B}{N}$ and $\mathrm{c}\equiv\frac{C}{N}$

Equation (27) represents Gibbs Duhem type of relation in scalable network queue. The variation in μ is not independent of the variations in θ and δ, but the variations of any one can be computed in terms of the variations of the other two. Hence from equation (27)

$$\frac{\partial\mu}{\partial\theta}=c\,. \tag{28}$$

$$\frac{\partial\mu}{\partial\delta}=b\,. \tag{29}$$

$$\frac{\partial\delta}{\partial\theta}=-\frac{c}{b}\,. \tag{30}$$

Gibbs Duhem law presents relation among the microscopic parameters of the queue in the differential form. We verify these differential relationships for both the universal queues M/M/1/B and M/M/C/C+B.

5 Conclusion and Future Work

We prove that microscopic parameters of scalable network queue are related to each other independent of macroscopic parameters of the queue. We derive equivalent Gibbs Duham law in scalable network queue. Our future work will be developing the techniques for calculating microscopic parameters and finding equivalent thermo dynamical laws applicable to any network queue.

References

[1] J.Y.Hui, Ezhan Karasan, “A Thermodynamic Theory of Broadband Networks with Applications to Dynamic Routing,” *IEEE J. Select. Areas Commun.*, vol. 13, 1995.

[2] H.B.Callen, “Thermodynamics and an Introduction to Thermostatics, second edition,” John Wiley &Sons, 1985.

Bandwidth Sharing of Low Priority Services for Efficient Call Admission Control in Cellular Networks

Kyungkoo Jun and Seokhoon Kang

Department of Multimedia System Engineering,
University of Incheon, Incheon, Korea

Abstract. We propose a novel call admission control scheme to support the multiservice of cellular networks particularly under heavy loaded cell condition. The key idea of our strategy is that the services which underutilize their assigned bandwidth because of their *think time* are dynamically grouped and forced to share the bandwidth in round robin especially when the resource availability is unfavorable. We present the simulation results that show our proposed scheme outperforms a conventional bandwidth reconfiguration strategy in terms of call blocking probabilities.

1 Introduction

We propose a novel call admission control scheme to remedy the bandwidth under–utilization problem of low priority services. In particular, even with the bandwidth adaptation capability [1][2][3][4][5], it is inevitable to allocate low priority services certain portion of bandwidth, however most of which are not fully utilized because of the traffic characteristic of the services though. For example, the most bandwidth allocated for WWW service is wasted because of the presence of user think time, the time period until clicking to access other pages

The key idea of our proposed strategy is that the services which do not fully utilize their assigned bandwidth are grouped into a set, and then forced to share the bandwidth in round robin with others in the same group rather than being allocated separate bandwidth, thus benefiting *bandwidth-efficient* services, which are generally real–time and, at the same time, have higher priority than others.

We assume in this paper the services belong to one of four classes of services which were recommended in UMTS domain as follows : *conversational class* (class 1) for voice or video conference traffic, *streaming class* (class 2) for real–time video streaming, *interactive class* (class 3) for WWW or database access, and *background class* for email or downloading. The class 1 services have the highest priority, while the class 4 the lowest.

This paper is organized as follows. Section 2 discusses our proposed call admission control strategy in detail and presents the algorithm flow chart. Section 3

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1379–1382, 2005.

shows the simulation results for the performance analysis of our proposed scheme. Section 4 concludes this paper.

2 The Proposed Call Admission Control Scheme

The proposed call admission control scheme is motivated by the fact that the traffic pattern of class 3 or 4 services, e.g., WWW surfing or database access contains a large portion of "think time". It is the time duration elapsed between the completion of the transfer of the previous request and the beginning of the transfer of the current request [6]. In our scheme, we exploit this wasted think time to benefit the admission of higher priority services, i.e., class 1 or 2. The proposed scheme is devised to target the situation when the cell is not able to accept incoming high priority services because of a large number of existing low priority services.

The scheme works as follows. When the class 1 or 2 services request the admission to a cell, but there is insufficient bandwidth remaining, the existing class 3 and 4 services hand over their bandwidth in order to accept those requests. Instead, they form a bandwidth–sharing group in which they are able to continue their services by sharing the bandwidth in round robin. The bandwidth allocated to the group is much smaller than the total sum of the bandwidth owned by each group members. Because of the intermittent presence of the think times, the actual bandwidth usage times of each service may not overlap so often, thus their services are able to proceed without noticeable service degradation even with the round robin sharing.

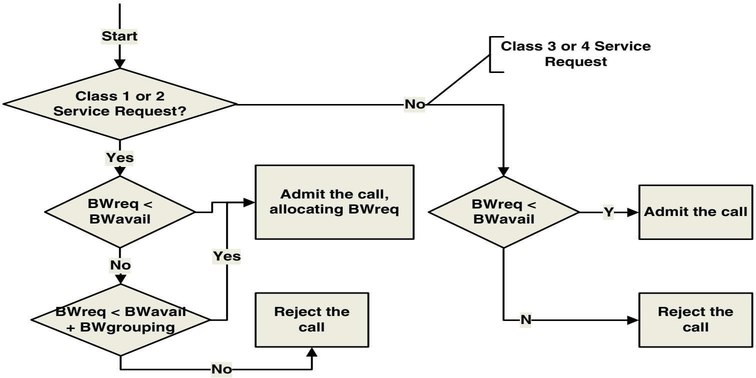

Start
Class 1 or 2 Service Request?
No
Class 3 or 4 Service Request
Yes
BWreq < BWavail
Admit the call, allocating BWreq
BWreq < BWavail
Y
Admit the call
No
Yes
BWreq < BWavail + BWgrouping
Reject the call
N
Reject the call
No

Fig. 1. The algorithm flow chart of our proposed scheme

Group Formation. The group formation proceeds in three steps. At first, the amount of bandwidth to be collected, $BW_{collected}$, is calculated as

$$BW_{collected} = BW_{requested} - BW_{available} \tag{1}$$

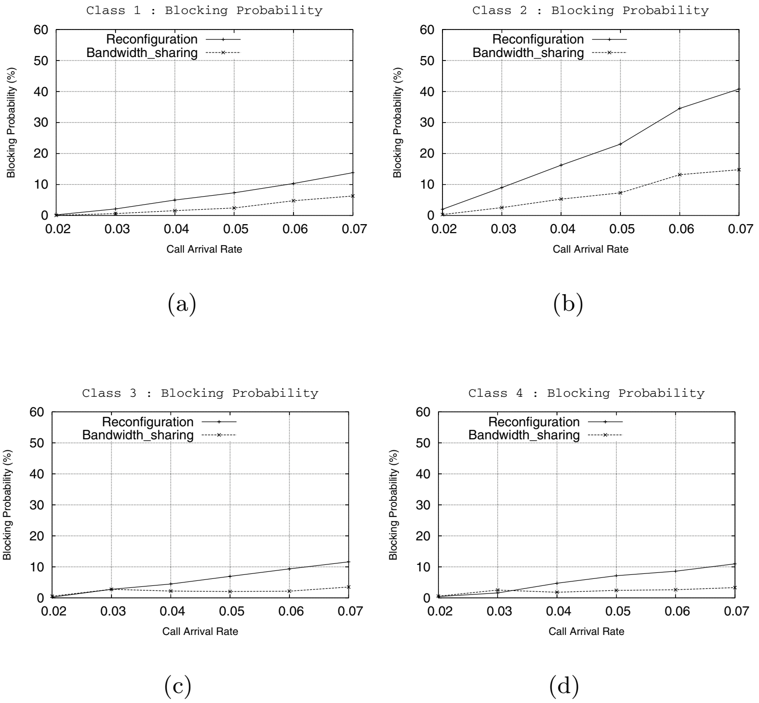
where $BW_{requested}$ is the bandwidth amount required by ingress high priority service, $BW_{available}$ is the bandwidth available in the cell. Secondly, it is determined which low priority services should be included in the group by deciding c in the below equation.

$$min\{c : \sum_{i=1}^{c} BW_i \geq BW_{collected}\} \tag{2}$$

where BW_i is the bandwidth owned by low priority service i. The group formation is aborted unless c satisfying the above equation is found. Finally, once c is determined, the group is formed by including $service_1$ to $service_c$. Then, the group is assigned with the bandwidth of which amount is

$$max\{BW_i : 1 \leq i \leq c\} \tag{3}$$

which ensures that, once a service in the group possesses the turn to use the bandwidth, it proceeds without latency caused by the lack of the bandwidth.



(a) (b)

(c) (d)

Fig. 2. Blocking probabilities of new and handoff calls (a) class 1 (conversational) (b) class 2 (streaming) (c) class 3 (interactive) (d) class 4 (background)

Figure 1 shows the flow chart of our proposed call admission control scheme. Note that our scheme is triggered only for the admission of class 1 and 2 services when there is insufficient bandwidth remaining in the cell.

3 Simulation and Performance Evaluation

We use a simulation map consisting of 19 hexagon-shaped cells, each of which has 2 Km radius, supporting up to 2 Mbps. The boundaries of the cells located at the borders of the map are wrapped around. As the radio propagation model in the simulation, we assume both the path loss and the shadowing model.

Mobile users in the simulation can have up to four services during their life time. The required bandwidth for each service class is: 16 Kbps, 32 Kbps, 8 Kbps, 16 Kbps for class 1, 2, 3, 4, respectively. The arrival rate λ of mobile hosts in the simulation follows the exponential distribution. The simulation results are collected for λ being incremented by 0.01 from 0.02 to 0.06.

Figure 2 compares the blocking probabilities, with the corresponding results of the reconfiguration strategy[1]. For all class services, the blocking probabilities were decreased, which demonstrates that our proposed scheme can admit more users than the reconfiguration strategy.

4 Conclusions

We proposed the call admission control scheme to prioritize class 1 and 2 services by having class 3 and 4 services be grouped into a bandwidth–sharing set particularly when the bandwidth of a cell is insufficient. Our proposed strategy was motivated by the fact that, in the case of interactive services like class 3 and 4, a large portion of assigned bandwidth is wasted because of the user think time. The simulation results were also presented.

References

1. Ye, J., Hou, J., Papavassiliou, S. : A Comprehensive Resource Management Framework for Next Generation Wireless Networks : IEEE Transactions on Mobile Computing, vol. 1, No. 4, 2002.
2. Meer, H., Corte, A., Puliafito, A., and Tomarchio, O. : Programmable Agents for Flexible QoS Management in IP Networks : IEEE J. Selected Areas in Comm., vol. 18, no. 2, Feb. 2000.
3. Solana, A., Bardaji, A., Palacio, F. : Capacity Analysis and Performance Evaluation of Call Admission Control for Multimedia Packet Transmission in UMTS WCDMA System : Proceedings of 5th IEE European Personal Mobile Communications Conference (EPMCC'03), 2003.
4. Chak, J., Zhuang, W. : Capacity Analysis for Connection Admission Control in Indoor Multimedia CDMA Wireless Communications : Wireless Personal Communications 12, 2000.
5. Xiao, Y., Chen, C. : QoS for Adaptive Multimedia in Wireless/Mobile Networks : Proceedings of 9th International Symposium in Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2001.
6. Liu, Z., Niclausse, N., Jalpa-Villanueva : Traffic model and performance evaluation of Web servers : Performance Evaluation vol. 46, 2001.

COPACC: A Cooperative Proxy-Client Caching System for On-demand Media Streaming*

Alan T.S. Ip[1], Jiangchuan Liu[2], and John C.S. Lui[1]

[1] The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{tsip, cslui}@cse.cuhk.edu.hk
[2] Simon Fraser University, Vancouver, BC, Canada
csljc@ieee.org

Abstract. Proxy caching is a key technique to reduce transmission cost for on-demand multimedia streaming. However, its effectiveness is limited by the insufficient storage space and weak cooperations among proxies and their clients. In this paper, we propose COPACC, a novel cooperative proxy-and-client caching system that combines the advantages of both proxy caching and peer-to-peer (P2P) client communications. We propose a comprehensive suite of protocols to facilitate the interactions among different network entities in COPACC. We also develop an efficient cache allocation algorithm to minimize the aggregated transmission cost of the whole system. Simulation results demonstrate that COPACC achieves remarkably lower transmission cost. Moreover, it is much more robust than a pure P2P system in the presence of node failures.

1 Introduction

Today's Internet has been increasingly used for carrying multimedia traffic, and on-demand streaming for clients is amongst the most popular networked media services. The limited server capacity, however, make efficient and scalable on-demand media streaming a challenging task. To reduce server/network loads, frequently used data is cached at proxies close to clients[1]. Streaming media, particularly those with asynchronous demands, could benefit with a significant performance improvement from proxy caching given their static nature in content and highly localized access interests. Another approach is to generalize the proxy functionalities into every client [2]. Such a P2P paradigm allows economical clients to contribute their storages for streaming. Video data originally provided by a server are spread among clients, thus amplifying the system capacity.

In this paper, we propose COPACC, a novel cooperative proxy-and-client caching system. We leverage the client-side caching to amplify the aggregated cache space and rely on dedicated proxies to effectively coordinate the communications. We develop an efficient cache allocation algorithm together with a comprehensive protocols suite to distribute video segments among the proxies

* This work is supported in part by the RGC Earmarked Grant.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1383–1387, 2005.

and clients such that the aggregated transmission cost is minimized. As most operations are executed by dedicated proxies, the system is resilient to client failures. We also embed an efficient indexing and searching algorithm for video contents cached across different proxies or clients. COPACC also makes effective use of multicast delivery, which further reduces the cost. The simulation results demonstrate that COPACC achieves remarkably lower transmission cost as compared to proxy-based caching with limited storage space. With the assistance from dedicated proxies, it is much more robust than a pure P2P system. Moreover, It scales well to larger networks, and the cost generally reduces when more proxies and clients cooperate with each other.

Fig. 1 depicts a generic architecture of COPACC. A cluster of proxies are logically connected to form overlay. The proxies and their clients are closely located with relatively low communication costs, while the proxies and the video server are located far away and incur higher costs. The video data are cached across proxies and clients of limited storage. As shown in Fig. 2, a video stream is partitioned into *prefix* and *prefix-of-suffix*. The proxies are responsible to cache the prefix of video, whereas the clients cache the prefix-of-suffix. Similar to [3], this setting helps to minimize the initial playback latency. When a client expects to play a video, it initiates a playback request to its home proxy, which intercepts the request and computes a streaming schedule. It then fetches the prefix, prefix-of-suffix, as well as the remaining part of suffix, and relays them to the client.

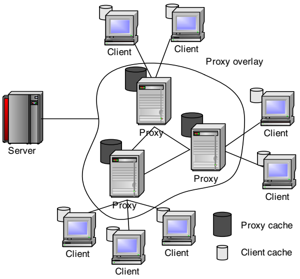


Fig. 1. The COPACC architecture

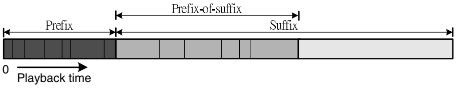


Fig. 2. Illustration of different portions of a video stream. The prefix is to be cached by proxies, while the prefix-of-suffix by clients

There are two key issues to be addressed: How to partition each video and allocate the prefixes and prefix-of-suffixes to different proxy and client? How to manage, search, and retrieve the cached data in different proxies and clients?

2 Optimal Cache Allocation Problem (CAP)

The optimal cache allocation problem (CAP) can be formulated as

$$\textbf{CAP} : \min \ Cost(\{p_j^i\}, \{q_{j,k}^i\}),$$

$$\text{s.t. } p_j^i, q_{j,k}^i \geq 0, j \in [1 \ldots H], k \in [1 \ldots K_j]; \quad \textstyle\sum_{i=1}^{N} p_j^i \leq s_j^p; \quad \sum_{i=1}^{N} q_{j,k}^i \leq s_{j,k}^c;$$
$$\textstyle\sum_{j=1}^{H} p_j^i + \sum_{j=1}^{H} \sum_{k=1}^{K_j} q_{j,k}^i \leq V^i,$$

where $Cost(\{p_j^i\}, \{q_{j,k}^i\})$ is the total transmission cost given prefix allocation $\{p_j^i\}$ and prefix-of-suffix allocation $\{q_{j,k}^i\}$; the second and third constraints follow the cache space limit of proxy j and that of client k of proxy j, respectively.

2.1 Single Proxy with Client Caching

We first consider a single proxy and multiple clients system. Since the transmission cost depends only on how the video are partitioned, we can assume that the caches of all the clients form an aggregated cache space, and derive the minimum transmission cost by finding the optimal values of $\{P^i\}$ and $\{Q^i\}$ subject to cache space constraints S^p and S^c. We define an auxiliary cost function $C^i(P^i, Q^i)$, which is the cost for delivering video i with prefix size P^i and prefix-of-suffix size Q^i. Note that $Cost(\{p_j^i\}, \{q_{j,k}^i\})$ is now equal to $\sum_{i=1}^{N} C^i(P^i, Q^i)$. The problem can then be solved by *dynamic programming.* It is applicable with arbitrary cost function $C^i(P^i, Q^i)$, which can be instantiated given a specific transmission scheme. As an example, assume both a server-to-client and a client-to-client transmissions are unicast-based and relayed by a proxy, $C^i(P^i, Q^i)$ can be derived as $\lambda f^i \cdot [w^{c \leftrightarrow p} P^i + 2w^{c \leftrightarrow p} Q^i + (w^{s \rightarrow p} + w^{c \leftrightarrow p})(V^i - P^i - Q^i) + w^{in}(P^i + Q^i)]$, where the first four terms in the square bracket represent the costs for retrieving prefix, prefix-of-suffix, the remaining suffix, and the internal cost of the proxy.

2.2 Multiple Proxies with Client Caching

We now consider the general COPACC system consisting multiple proxies and their respective clients. It involves interactions among several proxies and clients, and the unit transmission costs for the proxy-to-proxy and client-to-proxy links can be heterogeneous. In fact, we formally prove that CAP is NP-hard in this general case (see [4]). We thus resort to a practically efficient heuristics, which consists of two phases: first, it partitions the prefix and prefix-of-suffix for each video; second, given the partitions, it allocates the segments of prefixes and prefix-of-suffixes to the proxies and clients.

1) Partitioning of prefix and prefix-of-suffix: In this phase, we approximate the system by a single proxy system with aggregated proxy cache space S^p and aggregated client cache space S^c. An approximate solution of $\{P^i\}$ and $\{Q^i\}$ can be directly obtained using the dynamic programming algorithm.

2) Allocation to proxy and client caches: In this phase, we further partition the prefix and prefix-of-suffix, and allocate them to the proxies and clients. Since the allocation for prefixes to proxy caches is independent from that for prefix-of-suffixes to client caches, we separate the two allocation problems and solve them individually. The optimal prefix allocation problem (**PA**) is formulated as

$$\begin{array}{l} \mathbf{PA} : \min \ \sum_{i=1}^{N} \sum_{j=1}^{H} W^p(i, j, p_j^i) \\ s.t. \ \ \sum_{j=1}^{H} p_j^i = P^i, \ i \in [1 \ldots N]; \ \ \sum_{i=1}^{N} p_j^i \leq s_j^p, \ j \in [1 \ldots H]. \end{array}$$

As $W^p(i, j, p_j^i)$ can be instantiated as $\sum_{j'=1}^{H} p_j^i [w_{j,j'}^{p \rightarrow p} + w_{j'}^{c \leftrightarrow p}] \lambda_{j'} f_{j'}^i$ for unicast delivery, **PA** can be relaxed as a linear programming problem by re-writing it to $\min \sum_{i=1}^{N} \sum_{j=1}^{H} \bar{W}^p(i, j) \cdot p_j^i$.

We should also consider the optimal suffix-of-prefix allocation problem (**SA**) for client cache. Obviously, both the problem **SA** and the cost function itself have similar structure as that of problem **PA**. Thus, we omit the derivation of **SA** here. Note that the linear programming relaxation also applies for **SA**. The optimizations shown above can also be applied to multicast delivery (see [4]).

3 The Cooperative Proxy-Client Caching Protocol

As shown in Fig. 1, COPACC operates as a two-level overlay, where the first level consists of all the proxies, and the second level consists of each proxy and its own clients. The interactions among different entities in this two-level overlay are specified by a cooperative proxy-client caching protocol, which consists of three subprotocols. 1) *Cache allocation and organization* protocol specifies the election of proxy coordinator, which executes the optimal cache allocation algorithm and disseminates the lookup information using simplest hashing. 2) *Cache lookup and retrieval* protocol defines the discovery and retrieval of cache between proxies. 3) *Client access and integrity verification* protocol performs verification operation, which detects forged video data through a simple yet effective signature-based verification algorithm. The details of this protocol suite can be found in [4].

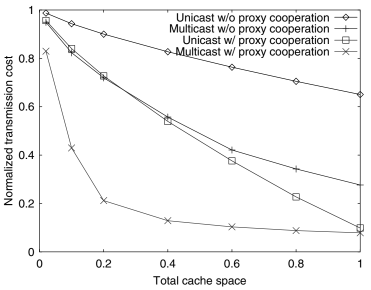


Fig. 3. Transmission cost as a function of the total proxy-client cache space

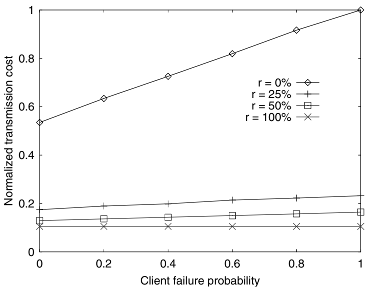


Fig. 4. Transmission cost versus client failure probability

4 Performance Evaluation

A primary design objective of COPACC is to reduce the transmission cost. Fig. 3 plots the transmission cost as a function of the total cache space, where the proxies and clients respectively contribute half of the total cache size. The cache sizes are normalized by the total size of the video repository, and the transmission costs are normalized by the corresponding cost of a system with no cache. Not surprisingly, increasing the total space reduces transmission cost. With unicast, the cost decreases linearly, while with suffix multicast, it decreases much faster. When the total cache space is 0.2, the cost with suffix multicast has been reduced to 0.2; in other words, a 20% cache space leads to a 80% cost reduction, which implies that batching the requests from local clients can avoid a significant amount of remote transmissions. It is also clear that the cost with cooperative proxies is much lower, particularly when multicast is also enabled in local paths.

The robustness in the presence of client failures is also a critical concern in COPACC. In Fig. 4, we show the transmission cost as a function of different client failure probabilities. We vary, r, the fraction of the total proxy cache space in the total cache space from 0% to 100%. When $r = 0\%$, COPACC degenerates to a pure P2P system, and, when $r = 100\%$, it degenerates to a pure proxy-based system. We can see that, when there is no client failure, the costs for different r are quite close if there are certain cache existed in proxies. More importantly, the cost of the pure proxy-based system remains unchanged when increasing client failures, and that for $0\% < r < 100\%$ is also very stable. For illustration, even if r is 25%, the transmission cost only slightly increases with an increase of failure probability; when the failure probability is 1, the cost remains a low as 0.22. To the contrary, the cost of the pure P2P system quickly increases and reaches 1 (the cost of a zero-cache system), when all clients fail. Such results demonstrate that the use of dedicated proxies with suffix batching remarkably improves the robustness and resilience of COPACC in the presence of client failures.

In summary, COPACC effectively utilizes the client-side cache space and realizes a low-cost video streaming system. Yet with the assistance of dedicated proxies, it is quite robust to accomodate dyanmic and even malicious clients. More results supporting such arguments as well as the behavior of COPACC under diverse network configurations can be found in [4].

References

1. Liu, J., Xu, J.: Proxy Caching for Media Streaming over the Internet. IEEE Communications (2004)
2. Cui, Y., Li, B., Nahrstedt, K.: oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. IEEE JSAC **22** (2004)
3. Wang, B., Sen, S., Adler, M., Towsley, D.: Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In: Proc. IEEE INFOCOM'02, NY (2002)
4. Ip, A.T.S., Liu, J., Lui, J.C.S.: COPACC: A Cooperative Proxy-Client Caching System for On-Demand Media Streaming. Technical Report (2004, CUHK, http://www.cs.sfu.ca/~jcliu/Papers/TR-COPACC-Final/TR-COPACC.pdf)

A Rule-Based Approach for Consistent Proportional Delay Differentiation (Extended Abstract)

Jianbin Wei and Cheng-Zhong Xu

Department of Electrical and Computer Engineering,
Wayne State University, Detroit, Michigan 48202
{jbwei, czxu}@wayne.edu

Abstract. Proportional delay differentiation (PDD) aims to maintain pre-specified packet queueing-delay ratios between different classes of traffic at each hop. Existing rate-allocation approaches for PDD services assume the average queueing delay of a class is inversely proportional to its service rate. This assumption may not be valid in a non-heavily loaded system. In this paper, we present a rule-based approach to provide *consistent* PDD services under various load conditions. In this approach, the service rate of a class is adjusted according to a set of control rules. Simulation results demonstrate that, in comparison with other rate-allocation approaches, the rule-based approach is able to provide PDD services under light, moderate, and heavy load conditions consistently.

1 Introduction

Proportional delay differentiation (PDD) service model is to maintain pre-specified queueing delay ratios between different classes [2]. In existing rate-allocation approaches for PDD services, the service rate of a class is adjusted periodically based on current system states [1, 2, 4]. For example, in backlog-proportional rate (BPR), a class's service rate is adjusted according to its backlogged queue length [2]; in joint buffer management and scheduling (JoBS), the service rate is set based on delay predictions of its backlogged traffic [4]. In [1], the authors proposed a linear feedback control approach (referred to as LFB in the rest of this paper), in which a class's service rate is adjusted according to the difference between its normalized head-of-line delay and the average of all backlogged classes. These approaches assume that the average queueing delay of a class is inversely proportional to its service rate. This assumption is valid only during busy periods of the system. These approaches thus are unable to deliver PDD services in a non-heavily loaded system because of the existence of idle periods. In this paper, we present a rule-based rate-allocation approach that relies on a general relationship between the amount of allocated rate and queueing delay to providing PDD services consistently.

2 Design of the Rule-Based Approach

The rule-based rate-allocation approach is based on fuzzy control theory to provide consistent PDD services under various workload conditions. Let W_i denote the average

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1388–1392, 2005.

delay of class i computed during a sampling period, and δ_i its pre-specified differentiation parameter. For class i and class j in the system, PDD requires W_i/W_j equals to δ_i/δ_j. In the rule-based approach, the service rate of a class in sampling period $k+1$, denoted by $u(k+1)$, is adjusted according to its error $e(k)$ (*i.e.,* difference between the target delay ratio and the achieved one) and change of error $\Delta e(k)$ in sampling period k. Such adjustment is controlled by a set of rules about heuristic knowledge described by fuzzy logic methods. Figure 1 shows the control structure of the rule-based approach. The rule-base contains quantified control knowledge about how to adjust a class's service rate according to the $e(k)$ and $\Delta e(k)$. The fuzzification interface converts controller inputs into certainties in numeric values of the "triangle" membership functions that are defined for the inputs. The inference mechanism activates and applies rules according to fuzzified inputs, and generates fuzzy conclusions for defuzzification interface. The defuzzification interface converts fuzzy conclusions into the change of service rate of a class in numeric value using "center average" method.

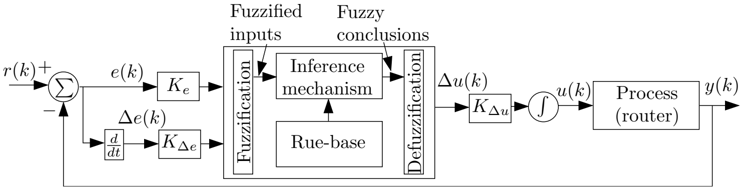
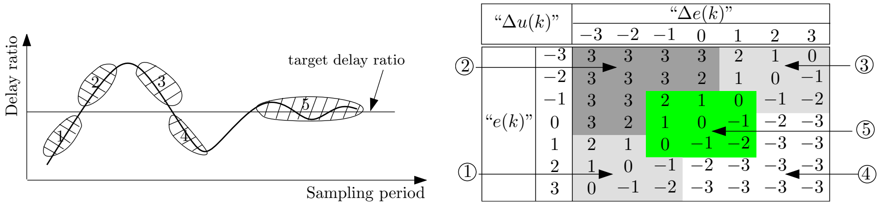


Fig. 1. The control structure of the rule-based approach

In the approach, class 1 is selected as the base class. A control loop is associated with every other class. In the control loop of class i, the reference input for kth sampling period $r_i(k)$ is $\delta_i(k)/\delta_1(k)$. The output of the loop is the achieved delay ratio of class i to the base class. The error $e_i(k)$ and the change of error $\Delta e_i(k)$ are therefore defined as $\delta_i/\delta_1 - W_i(k)/W_1(k)$ and $e_i(k) - e_i(k-1)$, respectively. The output of the controller is $\Delta u_i(k)$, the rate adjustment of class i. As shown in Figure 1, there are three control factors in the controller. Thus, the inputs of the controller are $K_e e(k)$ and $K_{\Delta e}\Delta e(k)$. The output $u(k) + K_{\Delta u}\Delta u(k)$ is a class's service rate for sampling period $k+1$.

The control rules is defined using linguistic variables and values from fuzzy control theory. The linguistic variables are used to describe each of the controller inputs and output. The linguistic variables assume linguistic values, which are represented using integers. We have "3" ("-3") describes positive (negative) large in size; "2" ("-2") represents positive (negative) medium in size; "1" ("-1") describes positive (negative) small; and "0" is zero in size.

We first analyze the effect of the controller on the achieved delay ratio in PDD service provisioning as illustrated in Figure 2(a). In this figure, five zones of different characteristics can be identified. Zone 1 and 3 are characterized with opposite signs of $e(k)$ and $\Delta e(k)$. That is, in zone 1, $e(k) > 0$ and $\Delta e(k) < 0$; in zone 3, $e(k) < 0$ and $\Delta e(k) > 0$. In these two zones, the error is self-correcting, and the achieved delay ratio is moving towards to the target one. Then $\Delta u(k)$ is set to either speed up or slow down the current trend. Zone 2 and 4 are characterized with same signs of $e(k)$ and $\Delta e(k)$. That is, in zone 2, $e(k) < 0$ and $\Delta e(k) < 0$; in zone 4, $e(k) > 0$ and $\Delta e(k) > 0$. In these two zones, the error is not self-correcting, and the achieved delay ratio is moving away from the target one. Then $\Delta u(k)$ is set to reverse the current trend. Zone 5 is related to situations characterized with rather small magnitudes of $e(k)$ and $\Delta e(k)$. That is, both $e(k)$ and $\Delta e(k)$ are close to zero, and the system is at a steady state. Thus $\Delta u(k)$ is set to maintain current state and correct small deviations from the target delay ratio.



"$\Delta u(k)$"		"$\Delta e(k)$"						
		−3	−2	−1	0	1	2	3
"$e(k)$"	−3	3	3	3	3	2	1	0
	−2	3	3	3	2	1	0	−1
	−1	3	3	2	1	0	−1	−2
	0	3	2	1	0	−1	−2	−3
	1	2	1	0	−1	−2	−3	−3
	2	1	0	−1	−2	−3	−3	−3
	3	0	−1	−2	−3	−3	−3	−3

(a) Illustration of control effect.

(b) The control rules.

Fig. 2. Design of control rules

The resulted control rules based on the analysis are summarized in Figure 2(b). A general linguistic form of these rules should read as: *If* premise *Then* consequent. For example, assuming that both $e(k)$ and $\Delta e(k)$ are "1", the rule then is: *If* the error is positive small *and* the change of error is positive small *Then* the change of service rate is negative medium. The detailed design is presented in [5].

3 Experimental Results

We compare the proposed approach with JoBS, BPR, and LFB. The experiments assumed two classes in the system. We assumed distributions of packet inter-arrivals and sizes similar to those in [2]. The packet inter-arrivals followed a Pareto distribution with the shape parameter to be 1.5. In these packets, 40% were 40 bytes, 50% were 550 bytes, and 10% were 1500 bytes. The target delay ratio δ_2/δ_1 was set to 4. The system load changed from 60% to 95% for every 200 sampling periods, which is set to 10,000 departed packets. The control factors of the controller were set as: $K_e = K_{\Delta e} = 0.1$ and $K_{\Delta u} = 1$. Figure 3 presents the experimental results.

First of all, we observe from the figure that the rule-based approach can provide consistent PDD services if the system load is no less than 70%. In comparison, none of the other rate-allocation approaches was able to achieve the goal. JoBS and BPR can guarantee PDD services only when the system load becomes as high as 90%. This is

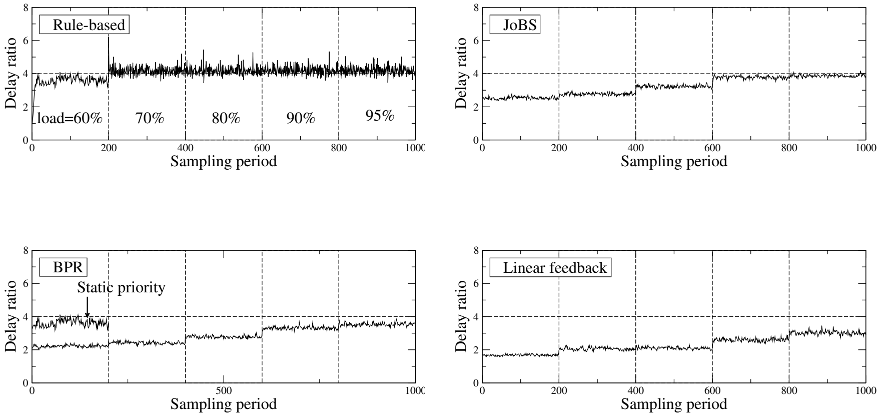


Fig. 3. Comparison with other rate-allocation approaches

because they adjusted a class's service rate according to its queueing delay. In systems with moderate load, the idle periods leads to a deviation of the achieved delay ratio from the target.

LFB fails to deliver required PDD services in all test cases. It is because the approach was mainly designed for differentiated service provisioning over busy periods only. In a time interval with mixed busy and idle periods, a small idle time percentage leads to a great performance deterioration. LFB is a rate-allocation approach with feedback control. It, however, adjusts a class's service rate according to the difference between its normalized head-of-line delay and the average of all backlogged classes. In contrast, the rule-based approach adjusts the service rate according to the error of achieved delay ratio. It approximates the nonlinear relationship between the queueing delay and the service rate by taking advantage of fuzzy control theory.

From the figure, we observe that when the system load becomes lower than 70%, the achieved delay ratio is always smaller than the target ratio. This infeasibility is possible for PDD services in a work-conserving system due to the constraints of conservation law [3]. We conducted an experiment using the static priority approach to obtain the upper bound of the feasible delay ratios for a system of 60% load. From the figure, it can be observed that the rule-based approach can closely approximate the upper bounds. The achieved delay ratios due to other approaches are much smaller than the bounds. This further demonstrates the superiority of the rule-based approach.

4 Conclusions

In summary, we have proposed a rule-based approach for PDD service provisioning in this paper. It adjusts rate allocation between different classes using a set of control rules that quantifies heuristic control knowledge that are valid under various load conditions. In comparison with other rate-allocation approaches, the approach is demonstrated to be effective for provisioning of consistent PDD services.

References

1. N. Christin, J. Liebeherr, and T. F. Abdelzaher. A quantitative assured forwarding service. In *Proceedings of IEEE Infocom*, volume 2, pages 864–873, June 2002.
2. C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of SIGCOMM*, pages 109–120, 1999.
3. C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1), 2002.
4. J. Liebeherr and N. Christin. JoBS: Joint buffer management and scheduling for differentiated services. In *Proceedings of IWQoS 2001*, pages 404–418, Karlsruhe, Germany, June 2001.
5. J. Wei and C. Xu. Consistent proportional delay differentiation: A fuzzy control approach. Technical report, Department of Electrical and Computer Engineering, Wayne State University, June 2004.

Extended Dominating Set in Ad Hoc Networks Using Cooperative Communication*

Jie Wu, Mihaela Cardei, Fei Dai, and Shuhui Yang

Department of Computer Science and Engineering,
Florida Atlantic University, Boca Raton, FL 33431

Abstract. We propose a notion of extended dominating set whereby each node in an ad hoc network is covered by either a dominating neighbor or several 2-hop dominating neighbors. This work is motivated by cooperative communication in ad hoc networks where transmitting independent copies of a packet generates diversity and combats the effects of fading. In this paper we propose several efficient heuristic algorithms for constructing a small extended dominating set.

1 Introduction

Dominating set (DS) has been widely used in ad hoc networks. A set is dominating if every node in the network is either in the set or a neighbor of a node in the set. When a DS is connected (i.e., its induced graph is connected), it is denoted as CDS. The problem of finding the minimum DS and minimum CDS is NP-complete. Many heuristic protocols have been proposed to find a minimal DS or CDS [1] [2] [3].

We propose a notion of *extended dominating set* based on *cooperative communication* (CC) [4]. CC makes single-antenna nodes in a multi-user scenario share their antennas to create a virtual multiple-input multiple-output (MIMO) system. CC can potentially combine the following two advantages: (1) the power savings provided by multi-hopping, and (2) the spatial diversity provided by the antennas of separate mobile nodes. In CC, transmitting independent copies of a packet generates diversity and combats the effects of fading. In this way, k copies of the same packet can potentially reach a receiver outside the normal transmission range without increasing transmit power. Under the CC model, a DS is called an extended dominating set (EDS) if, for every node in the network, it is in the set, it has a neighbor in the set, or it has k 2-hop neighbors in the set. In Fig. 1 (a), $\{u, v, w\}$ forms a CDS. If using CC, and $k = 2$, node x is covered by two 2-hop neighbors, u and v. Then, w can be withdrawn and $\{u, v\}$ forms an EDS. Since the set is connected, it is also called an *extended connected dominating set* (ECDS). Later, we will define weakly connected EDS (EWCDS). In EWCDS, the broadcast will be successful for at least one source in EDS; whereas in the ECDS the broadcast will be successful for any source in the EDS. In Fig. 1 (b), $\{u, v, x\}$ forms an EWCDS for $k = 2$ since x can retrieve the complete packet when either u or v is the source, while neither u nor v can when x is the source.

* This work was supported in part by NSF grants ANI 0073736, CCR 0329741, CNS 0422762, CNS 0434533, and EIA 0130806. Contact address: jie@cse.fau.edu.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1393–1396, 2005.

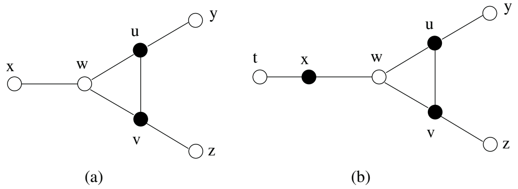


Fig. 1. (a) A sample with CDS: $\{u, v, w\}$ and ECDS: $\{u, v\}$. (b) EWCDS: $\{x, u, v\}$.

Wu and Lou [5] classified CDS formation methods into *global, quasi-global, quasi-local*, and *local* depending on the amount of information each node has, and the complexity of both message and time to determine a CDS. This paper focuses on some non-trivial extensions of various methods for ECDS/EWCDS and proposes (1) global solutions for EWCDS, (2) quasi-global solutions for EWCDS, (3) quasi-local solutions for EDS and ECDS, and (4) local solutions for EDS and ECDS. For more technical details, theorem proofs and simulation results, readers are refereed to [6].

2 Extended Dominating Set

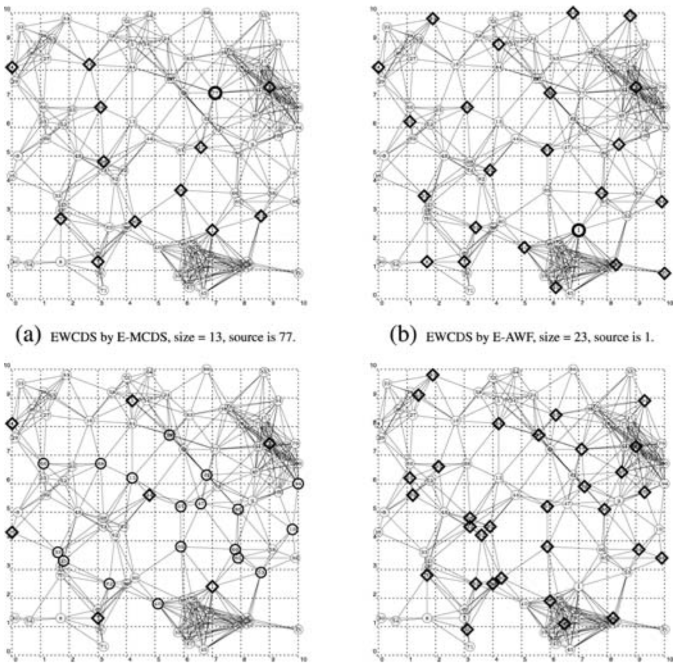
Given a set V of points in a 2D space, a normal transmission range r, and a CC range $r^{'}$, we define a graph with vertex set V and an arc from vertex v to vertex u iff the Euclidean distance, $d(v, u)$, is no more than r. In addition, we define a quasi-arc from vertex v to vertex u iff $r < d(v, u) \leq r^{'}$. When $r^{'} = 2r$, the corresponding graph can be approximated by a single unit disk graph, where a quasi-arc exists between any two vertices (called quasi neighbors) that are separated by two hops.

Definition 1. *A subset of nodes is an EDS if every node is (a) in the subset, (b) a regular neighbor of a node in the subset, or (c) a quasi neighbor of k nodes in the subset.*

Definition 2. *An EDS is* strongly connected *under the CC model (denoted as ECDS) if for any node u in the set sending a packet, the packet should be fully received by all other nodes eventually. Only nodes with a fully received packet (including u) are able to forward the packet once.*

If the connectivity condition holds for at least a particular node u, it is called *weakly connected* (EWCDS). It is known that DS and CDS problems in unit disk graphs are NP-complete. We proved that EDS, ECDS, and EWCDS problems are NP-complete.

The ECDS/EWCDS can be used as a virtual backbone under the CC model. Such a backbone can support an efficient broadcast process and reduce searching space. Unlike broadcasting using regular DS, the source node may need a relay node (not in the EDS) to forward the packet to a node in the EDS; otherwise, only the nodes in the EDS need to forward the packet.



(a) EWCDS by E-MCDS, size = 13, source is 77.

(b) EWCDS by E-AWF, size = 23, source is 1.

(c) ECDS by E-Cluster-LMST, size = 25.

(d) ECDS by E-Rule K, size = 31.

Fig. 2. Sample ECDS or EWCDS in an ad hoc network with 100 nodes

3 Heuristic Solutions

Global solutions for EWCDS. We consider a centralized greedy solution called extended MCDS (E-MCDS), based on Guha and Khuller's MCDS [1]. MCDS "grows" a tree from a selected root until all nodes are covered. Non-leaf nodes form a CDS. We introduce the notion of *contribution* here: Each forward node contributes 1 to all its neighbors and $1/k$ to all quasi neighbors. The *reception ratio* of a node is the combined contribution of its forward (quasi) neighbors. The algorithm is to find a minimum EWCDS so that all other nodes are *reachable* (i.e., each node has a reception ratio of at least 1). To ensure a constant approximation ratio, we employ a *mutual exclusion rule* which uses the concept of independent set. Fig. 2 (a) shows the EWCDS generated by the E-MCDS in a random, 100-node connected graph.

Quasi-global solutions for EWCDS. We extend the AWF algorithm [2] for CDS. AWF contains topology sorting, sequential clustering, and gateway designation procedures. In our extended AWF algorithm for EWCDS (E-AWF), we modify the gateway designation procedure to use an extended gateway designation approach, thus the set of selected nodes becomes an EWCDS, and the algorithm has a constant approximation ratio. Fig. 2 (b) shows the EWCDS generated by E-AWF.

Quasi-local solutions for EDS and ECDS. We use the clustering approach as the solution for EDS and ECDS. By a quasi-local solution, we mean the solution completes with a high probability in a small number of rounds with an occasional large number of rounds for completion. The clustering algorithm contains the selection of clusterheads and gateways. In our extended clustering approach (E-Clustering), each node operates on its 2-hop neighborhood. When a clusterhead is chosen, it dose not only contribute 1 to the coverage of its neighbors, but also $1/k$ to its quasi neighbors. To extend EDS to ECDS, we use an extension of the local minimum spanning tree (LMST) algorithm [7] to select gateways, whereby the 1-hop neighborhood includes the current clusterhead, all clusterheads within 5 hops, and their pairwise minimum "virtual path" in terms of hop count. In this way, each pair of neighboring clusterheads has a virtual link and LMST can be applied. The EDS generated by extended clustering and gateway nodes together forms an ECDS that has a constant approximation ratio. In Fig. 2 (c), the clusterheads are noted by diamonds, and the gateways by bold circles.

Local Solutions for EDS and ECDS. In local backbone construction, each node maintains only 2-hop information and performs: (1) Dai and Wu's pruning rule [3] (Rule K) for constructing a CDS, and (2) an aggressive pruning rule to remove nodes from the CDS while still maintaining local coverage and connectivity. We develop the extended Rule K (E-Rule K), which uses 2-hop information, including the markers of all 2-hop neighbors. A marked node u can be unmarked if all its 2-hop neighbors, regular and quasi, can be covered (including contribution cumulation) by other marked nodes with higher priority (noted as C) in the neighborhood, and the corresponding condition is called the *coverage condition*. The set derived by the pruning rule based on the coverage condition forms an EDS. To ensure connectivity, we require C to be connected under the CC model, the *connectivity condition*. We call C an *extended component* if it is strongly connected (based on Definition 2). A pruning rule that meets coverage and connectivity conditions preserves an ECDS with the expected size $O(1) \cdot |ECDS_{opt}|$, where $ECDS_{opt}$ is an optimal solution to the ECDS problem. Fig. 2 (d) shows the ECDS generated by the E-Rule K.

References

1. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica 20 (1998) 374387
2. Alzoubi, K.M., Wan, P.J., Frieder, O.: Distributed heuristics for connected dominating set in wireless ad hoc networks. Journal of Communications and Networks 4 (2002) 2229
3. Dai, F., Wu, J.: An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. IEEE Transaction on Parallel and Distributed Systems 15 (2004) 908920
4. Nosratinia, A., Hunter, T.E., Hedayat, A.: Cooperative communication in wireless networks. IEEE on Communications 42 (2004) 7480
5. Wu, J., Lou, W.: Forward node set based broadcast in clustered mobile ad hoc networks. Wireless Communications and Mobile Computing 3 (2003) 141154
6. Wu, J., Cardei, M., Dai, F., Yang, S.: Extended dominating set and its applications in ad hoc networks using cooperative communication. (submitted for publication)
7. Li, N., Hou, J., Sha, L.: Design and analysis of an MST-based topology control algorithm. In: Proceedings of INFOCOM. (2003)

INTERMON: An Architecture for Inter-domain Monitoring, Modelling and Simulation

Elisa Boschi[1], Salvatore D'Antonio[2], Paul Malone[3], and Carsten Schmoll[1]

[1] Fraunhofer FOKUS, Kaiserin Augusta allee, 31 10589 Berlin, Germany
{elisa.boschi, carsten.schmoll}@fokus.fraunhofer.de
[2] Lab. ITEM – Consorzio Interuniversitario Nazionale per l'Informatica (C.I.N.I.)
Via Diocleziano, 328 80125 Napoli, Italy
salvatore.dantonio@napoli.consorzio-cini.it
[3] Telecommunications Software and Systems Group, Waterford Institute of Technology,
Cork Road, Waterford, Ireland
pmalone@tssg.org

Abstract. In this paper we present a flexible architecture providing a communication platform between heterogeneous components for inter-domain QoS and traffic analysis in large-scale, multi-domain Internet infrastructures.

1 Introduction

Performing traffic analysis and guaranteeing inter-domain Quality of Service, or QoS, in large-scale, multi-domain Internet infrastructures is a challenging task. It involves performing real traffic measurements, analysing that traffic, ideally combining that data with routing information to match it to the network structure.

INTERMON provides different domains with a common infrastructure that offers several facilities used to gain a complete knowledge of network status as well as to perform QoS analysis in an inter-domain environment. For the user, INTERMON provides modelling, simulation and visualisation capabilities. In particular, a QoS modelling toolkit supports the analysis of network behaviour and various simulation techniques are used to study performance by proposing "what if" scenarios for traffic and topology. A visualisation component performs filtering, mapping and rendering to display resources and traffic flows for QoS planning.

2 The INTERMON Architecture

The INTERMON architecture specifies the interaction of the system components. It allows the system to monitor the current state of processing of measurement and simulation tasks and to store that state so that it is accessible to tools and users.

Figure 1 shows the logical "intra-domain" architecture setup (i.e. these components are present in each domain) highlighting the four layers of which it is composed: user interface, central control and storage, tools adaptation, and the tools layer. The architecture is built around a central server component called a Global Controller (GC).

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1397 – 1400, 2005.

The Global Controller's main role is to coordinate interaction between the attached components such as clients, visualisation, measurement and simulation and to maintain the status of currently active tasks.

For communication between Autonomous Systems (ASes) the GC integrates methods for Authentication, Authorization and Accounting -secured communication [1]. More details on the inter-domain communication and the architecture prototype implementation are to be found respectively in [2] and [4].

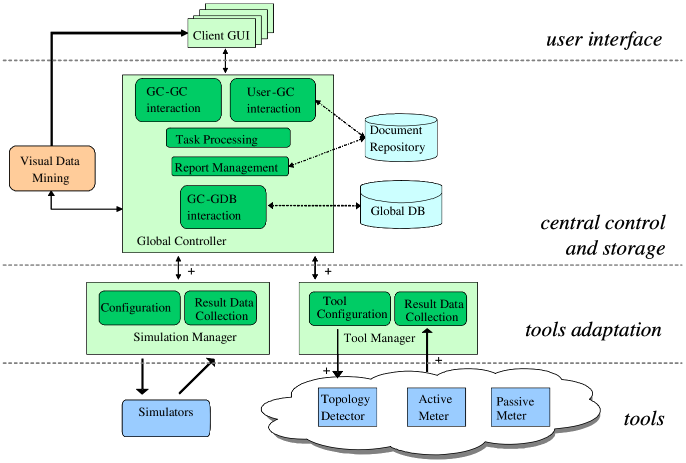


Fig. 1. The INTERMON Architecture

3 Scenario 1: SLA Violation Analysis

In this scenario a violation of an SLA has occurred and has been reported to the provider by the customer.

1. The provider performs a topology discovery for the period in which the violation occurred. The INTERMON database maintains a history of BGP-4 routing data and can be queried for the topology at the time of the violation. The result of such a query is an XML representation of the BGP topology and the visualisation of the topology showing the autonomous systems involved as well as whatever border routers have advertised their updates to the external repository. All of these tasks are performed through the use of the INTERMON tool *InterRoute*.
2. The *QoS Pattern Analyser* is used to compare end-to-end QoS data measured at the time of the violation (if available) with the violation and the BGP-4 topology. It has been shown in the project that there can be a relationship between frequent route changes (which can be seen by observing BGP-4 updates) and degradation in QoS measurements.
3. The provider can use the visualised topology obtained in step 1 to configure a monitoring probe called *MRCollector*, which can help the provider discover monitoring probes in the network. If a probe exists on the queried network element then

a list of interfaces, which can be monitored on the element, is returned to the INTERMON GUI. The provider can then use the INTERMON GUI to configure a "campaign" to monitor activity on that interface.

4. QoS/Traffic measurements are evaluated. Through the integration of the above-described tools it is possible for the provider to gain an insight into the behaviour of the network to aid in identification of the cause of the reported violation.

4 Scenario 2: "What If" Analysis

Modelling and simulation of network configurations allows operators to conduct "what if" type analyses of current or planned systems. Four separate inter-domain network simulators were developed and integrated into the INTERMON architecture (Cf. [3]). In order to validate them, a test bed was set up as in Fig 2 and a comparison of simulation results was made with the actual measured results from the test bed.

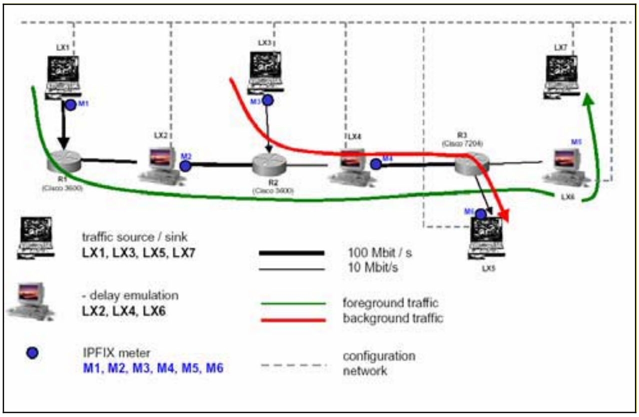


Fig. 2. The INTERMON test bed

The test bed consists of three Cisco routers, three Linux PCs configured to emulate network delay and four Linux end systems for traffic sources and sinks. The basic scenario is that delay sensitive foreground traffic is transmitted from LX1 to LX7. While this is happening, background traffic is transmitted from LX3 to LX5. Depending on the amount of traffic from LX3 to LX5 a bottleneck is created. The foreground traffic for this control scenario was 10 G.711 VoIP flows with a payload of 160Bytes every 20ms on each flow. The background traffic is high quality MPEG-4.

To start the simulation through the INTERMON GUI, the user must select the simulation tool to be used, choose the BGP topology representing the test bed between a set of previously stored ones, specify parameter values. When the simulation job is complete the user can retrieve a visualisation of the result by requesting this from the Visual Data Mining Module where filters are applied to the data and it is transformed to visual format.

For each of the implemented simulators, the results were compared with the measured data described above. One of these comparisons for the NS2-Hybrid simulator is shown in Fig. 3. A comprehensive evaluation of the individual simulation approaches is provided in [5].

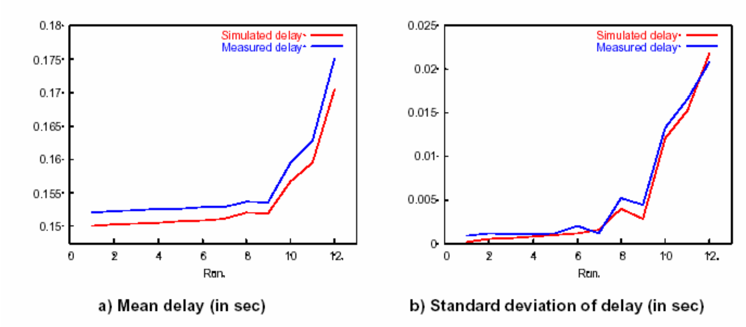


Fig. 3. Comparison of testbed measurements with NS2 Hybrid simulation

5 Conclusions

In this paper we have presented the INTERMON architecture, which integrates tools to study the impact of traffic and topology for end-to-end QoS in an inter-domain environment. The system comprises active and passive meters, topology detection components, and a modelling and simulation toolkit. The INTERMON system has been implemented, and subsequently validated through the scenarios shown in this paper.

Acknowledgements

This work has been performed under the partial financial support of IST project INTERMON IST-2001-34123, http://www.ist-intermon.org

References

[1] AAA, IETF web page: http://www.ietf.org/html.charters/aaa-charter.html

[2] Elisa Boschi, Salvatore D'Antonio, Giorgio Ventre, «Inter-domain Communication and Data Exchange», In Proceedings of IPS 2004, Budapest, Hungary, March 2004.

[3] INTERMON Deliverable 11: "Integration of the Inter-Domain Modelling and Simulation toolkit", 2003, InterMon project homepage: http://www.ist-intermon.org

[4] INTERMON Deliverable 15: "Final Architecture Specification", 2004

[5] INTERMON Deliverable 19: "Evaluation of Inter-Domain QoS Modelling, Simulation and Optimization", 2004

An UNA-Based Approach to Support Mobility in the Internet

Mahnhoon Lee[1] and Yongik Yoon[2]

[1] Department of Computing Science, Thompson Rivers University,
900 McGill Rd, Kamloops, BC, V2C5N3 Canada
mlee@cariboo.bc.ca
[2] Department of Multimedia Science, Sookmyung Women's University,
53-12 Chungpadong 2Ka, Yongsanku, Seoul, 140-742 Korea
yiyoon@sookmyung.ac.kr

Abstract. Mobile IP has been developed to give mobile nodes freedom to move to new locations away from home networks. However, Mobile IPv4 requires the implementation of address binding on all IPv4 hosts, i.e., modification, to prevent the long delay due to the triangular routing problem. The host modification is a huge hindrance to the deployment of mobility support. Meanwhile, a new scalable interoperability architecture called UNA (Universal Network Adaptation), which helps the Internet smoothly evolve to MTI (Multi-Tier Internet) or permits easy deployment of a new network technology, was presented. In this paper, we extend UNA, so that we can move the point of address binding from IPv4 hosts to UNA gateways. Therefore, mobility support with no triangular routing problem can be achieved in MTI, without host modification.

1 Introduction

IPv4 started up without any consideration of mobility support. Mobility support, MIPv4 (Mobile IPv4) [5] and MIPv6 (Mobile IPv6) [1], was recently considered and has been developed. When a mobile node (MN) moves into another new location and is assigned a new address, the address binding to the new address should occur to keep the on-going communications between the MN and the corresponding nodes (CNs). The address binding could take place at CNs or at a home agent (HA) of the MN, which provides CNs with the location service for MNs. When the address binding takes place at a HA, the traffic route between the MN and the CN becomes longer, and it may cause inability to provide good quality streaming services. The problem is called the triangular routing problem.

However, almost all IPv4 hosts, i.e., CNs, on the Internet do not support address binding, because IPv4, unlike IPv6, started without consideration of mobility support. Therefore, in order to solve the triangular routing problem in MIPv4, the address binding should be implemented on every IPv4 host, as explained in [6]. But the deployment of address binding on IPv4 hosts has been delayed, because the modification of hosts is required. As a result, mobility support in the Internet has not been used widely.

Meanwhile, UNA [4] was presented to provide accessibility from IPv4 realms to any other realm, such as IPv6, MTAII [2] and private IPv4. UNA reorganizes the

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1401 – 1404, 2005.

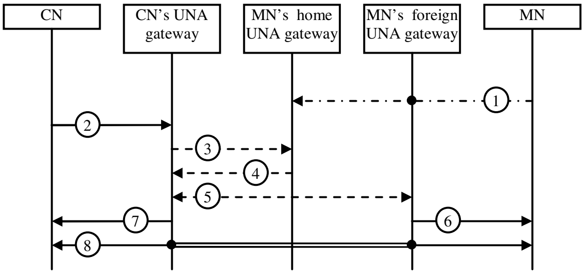
subscriber networks of the current IPv4 Internet into LASes (Local Autonomous System) by connecting them to the core network infrastructure through UNA gateways. Later, in [3], UNA was extended in MTI to provide the interoperability among any kind of LASes. Any IPv4-friendly realm, such as private IPv4, public IPv4, IPv6 realms, could be organized as LASes. Hosts in two different LASes can communicate with their own internetworking protocol, IPv4 or IPv6, without any host modification.

In this paper, we explain how to place the point of address binding at UNA gateways rather than hosts, so that we can solve the triangular routing problem without any modification of hosts for easy deployment of mobility support.

2 UNA-Based Mobility Support

2.1 When a CN Initiates a Communication to a MN

When a CN initiates a communication, a UNA tunnel is established between LASes. (Refer to [3].) If the peer host is a MN, then the home UNA gateway of the MN sends a bind update message to the UNA gateway of the CN at steps 3 and 4 in Figure 1. At step 5, the UNA gateway of the CN and the foreign UNA gateway of the MN exchange network specific addresses for the CN and the MN, as shown in [3]. At step 6, the foreign UNA gateway of the MN sends to the MN the global address of the CN's UNA gateway and the network specific address for the CN. Then the MN keeps the addresses for a while for the later handoff. At step 7, the CN receives the network specific address for the MN, as shown in [3]. Then, the CN and the MN can exchange traffic with the network specific addresses decided in the previous steps, as shown in [3].



Messages:
1: Registration of a new location, with a global address of the MN's current UNA gateway and the MN's current address in the foreign network
2: Initiation of user traffic (Refer to [3].)
3: UNA setup request
4: Bind update, with the global address of the MN's foreign UNA gateway and the MN's current address in the foreign LAS
5: UNA setup request and response
6: Notification of UNA setup, with the global address of the CN's UNA gateway and the network specific address for the CN (The MN keeps the global address of the CN's UNA gateway and the network specific address for the CN.)
7: The network specific address for the MN
8: User traffic (Refer to [3].)

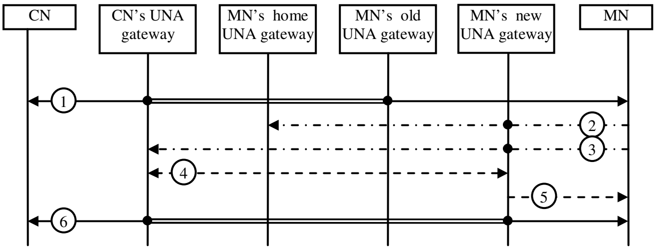
Fig. 1. When a CN initiates a communication to a MN

2.2 When a MN Initiates a Communication to a CN

This case is very similar to the previous case in Figure 1. The only difference is that when the CN is also a MN, a notification of UNA setup is also sent to the CN from the UNA gateway of the CN.

2.3 Handoff: When a MN Moves into Other LASes

In Figure 2, at step 2, the MN registers its new location to its home UNA gateway when it moves into other LAS. At step 3, the MN sends a bind update message with the previous and new location information, i.e., the previous and new global addresses of the MN's UNA gateways, and the previous and new address of the MN. (Actually the MN can communicate with several CNs in the same LAS. In that case, the MN sends a bind update message with the information for all CNs. Steps 4 – 6 are also applied to all CNs.) At step 4, two UNA gateways decide new network specific addresses for the MN and the CN. At step 5, the new network specific address of the CN is sent to the MN. At step 6, then the UNA gateway of the CN performs address binding between the old and new network specific addresses for the MN.



Messages:
1: User traffic
2: Registration of a new location of the MN
3: Bind update, with the previous and the new locations of the MN
4: UNA setup request with the notification indication and response (Note that the host specific address in the MN's new UNA gateway is changed.)
5: Notification of UNA setup, with the CN's UNA address and the host specific address for the CN (The MN updates the host specific address for the CN.)
6: User traffic

Fig. 2. When a MN moves into another LAS while it is communicating with a CN

2.4 When a CN Initiates a Communication to a MN, with an Incorrect Network Specific Address for the MN

The network specific address for the MN will be kept in the LAS of the CN for a while, even after the communication between the CN and the MN in Figure 1 finishes. Later the CN or any other host in the LAS can initiate a communication to the MN with the same network specific address for the MN. However the MN would have moved into other LAS so that the previous foreign UNA gateway no longer has the MN's information for the network specific address.

In the above scenario, the traffic from the CN will come to the old foreign UNA gateway of the MN. The old foreign UNA gateway of the MN sends a bind update message to the UNA gateway of the CN. The UNA gateway of the CN decides a new network specific address for the MN as shown at steps 3 - 6 in Figure 1. Then the UNA gateway of the CN performs address binding between the old and new network specific addresses for the CN until the old network specific address expires.

3 Concluding Remarks

We extend UNA in this paper to move the point of address binding for mobility support in MTI to UNA gateways from hosts or home agents, so that we could solve the triangular routing problem, with a few additional signals. Furthermore, UNA does not require any modification of hosts, and consequently neither does the extension of UNA, allowing for easy deployment.

The experimental implementation of the extension of UNA in MTI to support mobility is on going. In the future, we are planning to investigate mechanisms of fast handoff and group mobility using UNA on MTI, applying to WLAN environments.

References

1. D. Johnson, C. Perkison and J. Arkko, "Mobility Support in IPv6," *Internet-Draft draft-ietf-mobileip-ipv6-24.txt* (work in progress), June 30, 2003
2. Mahnhoon Lee, "MTAII: New Foundation for the Next Generation Internet," *IEEE GLOBECOM*, December 2003
3. Mahnhoon Lee and Y. Yoon, "MTI: Integration of UNA and MTAII for the Next Generation Internet", *EuroNGI NGI*, April 2005
4. Mahnhoon Lee and C. Han, "UNA: A Provision of Accessibility from the IPv4 Internet to Other Networks", *IEEE CCNC*, January 2005
5. C. Perkins, "IP Mobility Support for IPv4," *Internet RFC 3344*, August 2002
6. C. Perkins and D. Johnson, "Route Optimization in Mobile IP," *Internet Draft draft-ietf-mobileip-optim-09.txt* (work in progress), 2000

QoS Scalable Tree Aggregation

Joanna Moulierac and Alexandre Guitton

IRISA/University of Rennes I, 35 042 Rennes Cedex, France
{joanna.moulierac, alexandre.guitton}@irisa.fr

Abstract. Some of the main reasons which prevents the deployment of IP multicast are forwarding state scalability and control explosion problems. In this paper, we propose an algorithm called Q-STA (QoS Scalable Tree Aggregation) which reduces the number of forwarding states by allowing several groups to share the same tree. Q-STA accepts groups only if there is enough available bandwidth. Q-STA accepts much more groups and performs faster aggregations than previous algorithms.

1 Introduction

The deployment of multicast is limited mainly due to multicast forwarding state scalability and control explosion problems [1, 2]. Tree aggregation is a recent approach to deal with these problems: several groups share the same delivery tree within a domain. Consequently, less trees are maintained in the network and the number of forwarding states in routers is decreased. Moreover, the overhead of control messages required for tree maintenance is reduced. Several algorithms have been proposed to perform tree aggregation [3, 4]. Algorithm Aggregated QoS Multicast (AQoSM), described in [5], is a framework to support QoS multicast. The goal of AQoSM is to aggregate groups to trees while respecting bandwidth requirements of groups. Some groups may be refused because of the limited capacity of links.

In this paper, we propose a new algorithm called QoS Scalable Tree Aggregation (Q-STA) based on the framework of AQoSM. Q-STA accepts much more groups than AQoSM by building efficient trees and performing better aggregations. Additionally, the aggregation of Q-STA is faster than with AQoSM. We describe Q-STA in Section 2. Then, in Section 3, we compare AQoSM and Q-STA by extensive simulations.

2 Q-STA Algorithm

In order to deal with bandwidth constraints, each group is assigned a bandwidth requirement and each link is assigned a limited bandwidth capacity. To accept a group g, the bandwidth available on the tree has to be sufficient, otherwise g is rejected. Our protocol Q-STA is an extension of our proposition STA [6]. Q-STA has better performance than AQoSM because of the following aspects: (i) Q-STA builds efficient trees by utilizing links with lowest load in order to aggregate

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1405–1408, 2005.

further groups to this tree and to balance the load, (ii) Q-STA aggregates groups to minimum cost trees in order to spare network resources and (iii) Q-STA performs fast aggregations by evaluating few trees each time a group arrives whereas AQoSM evaluates all the trees in the multicast tree set.

Each time a new group g arrives, Q-STA considers only a subset of the multicast tree set (iii). This subset corresponds to the trees of cost between $(|g|-1)$ and $c(t_g)$, where $|g|$ is the number of members of g and $c(t_g)$ is the cost of the native tree of g. As soon as a candidate with enough bandwidth available is found, it is chosen for aggregation of g. As trees are evaluated in increasing order of their costs, the chosen tree minimizes the number of links used for the group g (ii). If no tree of adequate cost or with enough bandwidth is found for g, a new tree is builded and added in T. This new tree maximizes the bandwidth available on links (i). The loaded links are avoided and Q-STA builds trees with links that are little utilized by AQoSM.

3 Simulation Analysis

We conducted several simulation experiments on the graph Abilene which contains 11 nodes and 14 edges. We choose to assign 1 Gb/s of the 10 Gb/s available as the capacity dedicated to multicast for each link of Abilene. In this way, we take into account the utilization of the links by unicast communications.

We compared three algorithms: PIM-SM, AQoSM (with its bandwidth threshold equals to 0) and Q-STA (the program of Q-STA can be found at [7]). PIM-SM neither performs tree aggregation nor avoid congested links. The groups and their bandwidth requirements were the same for the three algorithms: 50% of the group requests were low-bandwidth (10 Kb/s), 30% of the group requests were medium-bandwidth (100 Kb/s) and the remaining 20% of the group requests were high-bandwidth (1 Mb/s). We implemented the node weighted model with 80% of the nodes having a weight of 0.2 and 20% of the nodes having a weight of 0.6 (see [8]). In this model, nodes with a large weight have an high probability of being members of groups. We varied the number of concurrent groups from $1,000$ to $10,000$. Each plot is an average of 100 simulation scenarios.

3.1 Number of Accepted Groups

Figure 1 shows the number of groups accepted by PIM-SM, AQoSM and Q-STA. These three algorithms accept the first $5,000$ groups because the network capacity is sufficient. After the first $5,000$ groups, PIM-SM refuses a large number of groups: it accepts only 800 of the next $5,000$ groups. Since PIM-SM does not take into account the available bandwidth of links, a group is refused as soon as its native tree uses saturated links. AQoSM accepts 200 more groups than PIM-SM. By building native trees maximizing the bandwidth available on links, Q-STA accepts $1,200$ groups more than AQoSM.

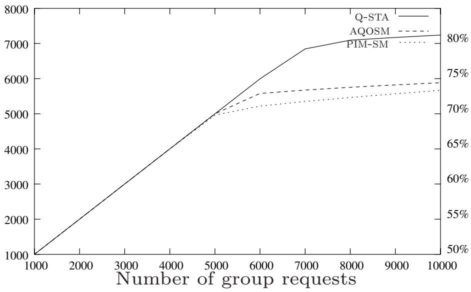


Fig. 1. Number of accepted groups.

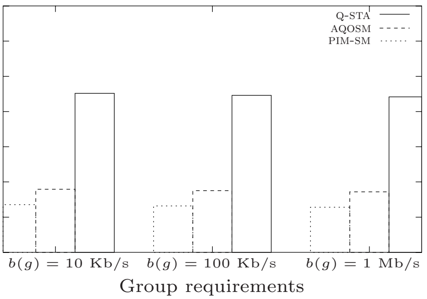


Fig. 2. Percent of accepted groups

3.2 Percent of Accepted Groups per Bandwidth Requirement

The number of accepted groups is not sufficient to evaluate an algorithm if groups have different requirements. Figure 2 plots the percent of accepted groups considering their bandwidth requirements. The three algorithms are fair: they accept groups independently of their bandwidth requirements. Unfair algorithms would have refused high-bandwidth groups in order to accept more low-bandwidth groups. Q-STA accepts 73% of high-bandwidth groups whereas AQoSM accepts only 59% and PIM-SM accepts 57% of these groups.

3.3 Aggregation Ratio

The aggregation ratio is defined as the number of trees over the number of concurrent groups. It determines the performance of a tree aggregation algorithm: a low aggregation ratio means that many groups have been aggregated. Figure 3 displays the aggregation ratio of AQoSM and Q-STA. The aggregation ratio of PIM-SM is not plotted since it is equal to 1. Before $6,000$ groups, AQoSM has a lower aggregation ratio than Q-STA: AQoSM builds less trees for the same number of groups. Instead of aggregating at all cost as AQoSM does, Q-STA balances the load by building more trees. After $6,000$ groups, Q-STA is slightly better than AQoSM, because Q-STA accepts more groups than AQoSM without building new trees. Q-STA builds around 50 more trees than AQoSM but handles around $1,200$ more groups. Finally, AQoSM and Q-STA build significantly less trees than PIM-SM (around 300 trees for AQoSM and Q-STA instead of $5,700$ trees for PIM-SM).

3.4 Number of Evaluated Trees

In AQoSM, all the trees are evaluated each time a new group arrives. In Q-STA, only trees that are likely to be candidates are evaluated. Additionally, Q-STA chooses as a tree for g the first tree with minimum cost that can cover g and that has enough bandwidth. Figure 4 shows the number of evaluated trees by AQoSM and Q-STA. Q-STA evaluates 45% less trees than AQoSM. To deal with

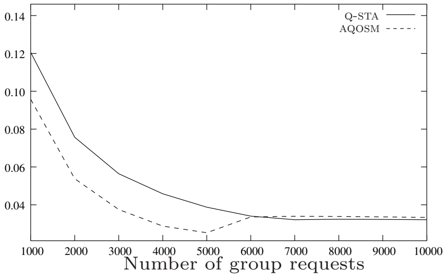


Fig. 3. Aggregation ratio

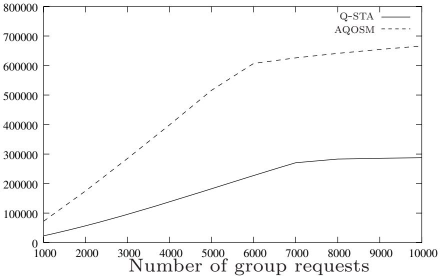


Fig. 4. Number of evaluated trees

10,000 group requests, AQoSM evaluates 380,000 trees more than Q-STA. This reduction of the number of evaluated trees enables faster aggregations.

4 Conclusion

In this paper, we proposed a new algorithm Q-STA for tree aggregation with bandwidth constraints. We compare the performance of our algorithm with the previous algorithm AQoSM by extensive simulations. While AQoSM accepts 200 groups more than PIM-SM, Q-STA accepts 1,400 more groups than PIM-SM. Indeed, Q-STA spares network resources by building efficient trees and by aggregating groups to low cost trees. Moreover, Q-STA builds as few trees as AQoSM and performs faster aggregations than AQoSM by evaluating 45% less trees.

References

1. Almeroth, K.: The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment. IEEE Network **14** (2000) 10–20
2. Wong, T., Katz, R.: An Analysis of Multicast Forwarding State Scalability. In: IEEE International Conference on Network Protocols (ICNP). (2000)
3. Cui, J.H., Kim, J., Maggiorini, D., Boussetta, K., Gerla, M.: Aggregated Multicast — A Comparative Study. In: IFIP Networking. (2002)
4. Liu, Z.F., Dou, W.H., Liu, Y.J.: AMBTS: A Scheme of Aggregated Multicast Based on Tree Splitting. In: IFIP Networking. (2004) 829–840
5. Cui, J.H., Kim, J., Fei, A., Faloutsos, M., Gerla, M.: Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks. In: IEEE Globecom. (2002)
6. Guitton, A., Moulierac, J.: Scalable Tree Aggregation with a Large Number of Multicast Groups. Internal publication 1663, IRISA (2004)
7. Q-STA: (2004) http://www.irisa.fr/armor/lesmembres/Guitton/recherche/qsta/qsta-en.html.
8. Fei, A., Cui, J.H., Gerla, M., Faloutsos, M.: Aggregated Multicast: an Approach to Reduce Multicast State. In: Global Internet Symposium. (2001)

Supermedia Transport for Teleoperations over Overlay Networks*

Zhiwei Cen[1], Matt W. Mutka[1], Danyu Zhu[1], and Ning Xi[2]

[1] Dept. of Computer Science and Engineering,
Michigan State University, East Lansing, MI 48824
{cenzhiwe,mutka,zhudanyu}@cse.msu.edu
[2] Dept. of Electrical and Computer Engineering,
Michigan State University, East Lansing, MI 48824
xin@egr.msu.edu

Abstract. In real-time Internet based teleoperation systems, the operator controls the robot and receives feedback through the Internet. Supermedia refers to robotic control commands, video, audio, haptic feedback, and other sensory information in the control system. Traditional transport services may not be able to meet the timely transmission requirements and dynamic priority changes of supermedia streams. Supermedia TRansport for teleoperations over Overlay Networks (STRON) uses multiple disjoint paths and forward error correction encodings to reduce end-to-end latency for supermedia streams. Network routes and encoding redundancy may be adjusted dynamically to meet the supermedia QoS requirements. NS2 simulations and evaluations using available bandwidth traces from globally distributed computing nodes show that STRON can significantly reduce latency compared with available transport services.

keywords: Teleoperation, Overlay networks, Forward error correction.

1 Introduction and Related Work

Teleoperation systems allow people to control automatic systems operating at remote sites. Traditional teleoperation systems use dedicated communication channels, which have higher costs and less flexibility. In an Internet based teleoperation system the operator manipulates a control device to issue task commands to the robot through the Internet. The feedback information, including video, audio and haptic information, enables the operator to be informed of the current state of the robot. We call all the information flowing in a teleoperation system supermedia [1]. Supermedia differs from traditional multimedia in that a larger variety of media are involved, and some media types are extremely sensitive to network delay. High latency may cause the operator to lose control of the remote robot.

* This paper was supported in part by NSF Grant No. 0334035.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1409–1412, 2005.

In this paper we present the Supermedia TRansport for teleoperations over Overlay Networks (STRON). STRON takes advantage of multiple disjoint overlay paths (such as RON [2]) and forward error correction encodings to improve the QoS performance. The networking routes and encoding redundancy may be adjusted dynamically to meet the QoS requirements of the supermedia streams. TCP Friendly Rate Control (TFRC) is used as the congestion control mechanism for each overlay connection, which ensures that the supermedia traffic is friendly to other network traffic. A more detailed version of the work can be found at [3].

2 System Architecture

The basic idea of STRON may be explained as follows. Each supermedia stream contains a series of messages generated by the teleoperation application in a variable or constant bit rate. The message is again chopped into packets with a certain size. Given p packets for a certain message that needs to be transmitted, STRON encodes the p packets into αp packets, where α ($\alpha \geq 1$) is the stretch factor. The encoded data packets are scheduled to be transmitted over multiple overlay paths, one of which is the IP path. As soon as the receiver collects $(1+\epsilon)p$ distinctive encoded data packets, the decoding algorithm can reconstruct the original data packets. Here ϵ is the reception overhead. The reception overhead is zero for some encoding algorithms and a small number for others. A class of erasure codes that has this property is called a digital fountain code [4]. By using a digital fountain code, the transport protocol can provide a rather reliable transport service without using acknowledgments and retransmissions.

The system architecture is shown in Figure 1. At the Sender-side Overlay Network Agent (SONA), the Disjoint Path Search Module (DPSM) runs a disjoint overlay network path searching algorithm through the connected overlay nodes. The Network Measurement Module (NMM) is usually deployed in overlay nodes as a common service. Some research efforts (such as *Pathload* [5]) have been dedicated to making accurate measurements of the network. The measurement results are fed into the Optimal Path Selection Module (OPSM).

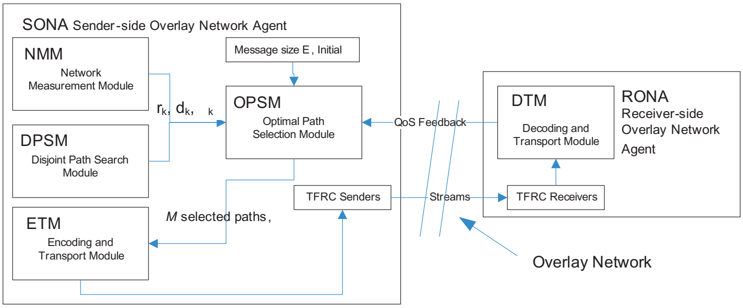

SONA Sender-side Overlay Network Agent
NMM
Network Measurement Module
DPSM
Disjoint Path Search Module
ETM
Encoding and Transport Module
Message size E, Initial
r_k, d_k, $_k$
OPSM
Optimal Path Selection Module
M selected paths,
TFRC Senders
QoS Feedback
Streams
DTM
Decoding and Transport Module
TFRC Receivers
RONA
Receiver-side Overlay Network Agent
Overlay Network

Fig. 1. The Architecture of STRON

The objectives of OPSM is to find the optimal disjoint path set to be used as active transmission paths and decide the amount of data sent over each active path. The design of the OPSM is discussed as follows. Suppose N disjoint (or semi-disjoint) overlay paths were found in DPSM. For each path k ($k \in [1, N]$), the following parameters are given: the single trip delay d_k in terms of seconds, the average throughput r_k in terms of bytes per second, and the packet loss rate β_k. The size of each message, which is the amount of effective data we need to transmit from the sender to the receiver, is E in terms of bytes. We need M disjoint paths out of the N given paths to minimize the latency between the sender and receiver. For each path i ($i \in [1, M]$) of the M paths, we also need to calculate V_i, which is the amount of data injected into this path by the sender. The system redundancy coefficient γ indicates the redundancy the system has over unexpected packet loss. Assume E_i is the effective data gathered from path i that is used in the data construction process, and we have $E_i = \frac{V_i(1-\beta_i)}{\gamma}$.

Assuming t is the time for the receiver to receive and reconstruct the original data, we have

$$V_i = (t - d_i)r_i \tag{1}$$

For all Paths, we have $\sum_{i=1}^{M} E_i = E(1+\epsilon)$ where ϵ ($\epsilon \in [0, 1)$) is the reception overhead of the digital fountain code, or $E = \frac{\sum_{i=1}^{M}[(t-d_i)r_i(1-\beta_i)]}{(1+\epsilon)\gamma}$. Solving t yields

$$t = \frac{E(1+\epsilon)\gamma + \sum_{i=1}^{M} r_i(1-\beta_i)d_i}{\sum_{i=1}^{M} r_i(1-\beta_i)} \tag{2}$$

To solve the problem, we may try set $s = \{(i_1, i_2, \cdots, i_M) | i_p \in [1, N], p \in [1, M]$, and for any $p \neq q, p \in [1, M], q \in [1, M], i_p \neq i_q\}$, which is a combination of set $[1, N]$ to minimize (2). The most straightforward method is to enumerate all the $\binom{N}{M}$ sets to find the subset s of $[1, N]$ that minimizes (2). When N is small, which is true in most cases, this method works well. The volume that needs to be sent over a selected disjoint path i, which is V_i, may be found by using (1). The stretch factor α of the digital fountain encoding may be calculated as $\alpha = \frac{\sum_{i=1}^{M} V_i}{E}$. A *transport plan* consists of a series of active overlay network paths, QoS parameters of these paths, the redundancy factor γ, stretch factor α and the number of packets (or volume of data) to be sent over each active overlay path. The Encoding and Transport Module (ETM) encodes the supermedia streams using α and passes the encoded packets to the overlay transport service. The stretch factor α is found by the Optimal Path Selection Module.

At the receiver side, the Decoding and Transport Module (DTM) of the Receiver-side Overlay Network Agent (RONA) decodes the packet streams received by the TFRC receivers and notifies the sender when enough packets have been collected for a certain message or a message is lost if a timeout occurs. The feedback to the sender also includes information telling the sender to what degree the redundancy in the encoding is effective so that the sender can adjust the redundancy level accordingly.

3 Simulation

To evaluate the effectiveness and efficiency of the system, an Optimal Path Selection Module was implemented in the application layer of the NS2 simulator. In order to simulate the behavior of the system under lossy network conditions, Poisson arrival process and Markov Modulated Poisson Process (MMPP) were used to simulate the packet loss process.

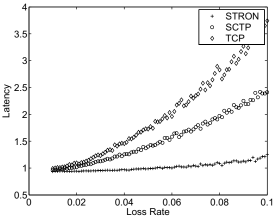


Fig. 2. Latency vs Loss Rate: STRON, TCP and SCTP (with 1 path)

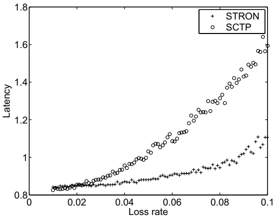


Fig. 3. Latency vs Loss Rate: STRON and SCTP (with 2 Paths)

Simulation	Average	STD
STRON 1p	3.9268	0.4947
STRON 2p	2.4604	0.5886
TCP 1p	6.0985	1.8439
SCTP 2p	4.0613	0.9387

Fig. 4. The Average and Standard Deviation of the Latencies

In the simulations, two candidate disjoint overlay network paths were established. Figure 2 shows the result of using one path. While the packet loss rate increases, STRON performs better than TCP and SCTP. In Figure 3 two paths were used for the transmission with STRON and SCTP. Although when the packet loss rate is low, SCTP performs a little better than STRON, STRON may take better advantage of the overlay paths as the loss rate increases. We also simulated the behavior of STRON using available bandwidth traces collected from globally distributed computing nodes. The average and standard deviation of the latency can be seen in Figure 4. We can see that STRON is also effective in reducing the variance of network latencies.

References

1. Fung, W.K., Xi, N., Lo, W.t., Song, B., Sun, Y., Liu, Y.h.: Task Driven Dynamic QoS based Bandwidth Allocation for Real-time Teleoperation via the Internet. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. (2003)
2. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient Overlay Networks. In: 18th ACM Symposium on Operating Systems Principles. (2001)
3. Cen, Z., Mutka, M., Zhu, D., Xi, N.: Supermedia Transport for Teleoperations over Overlay Networks. Technical Report MSU-CSE-05-01, Department of Computer Science and Engineering, Michigan State University (2005)
4. Byers, J.W., Luby, M., Mitzenmacher, M., Rege, A.: A Digital Fountain Approach to Reliable Distribution of Bulk Data. In: SIGCOMM. (1998)
5. Jain, M., Dovrolis, C.: Pathload: A Measurement Tool for End-to-end Available Bandwidth. In: Proc. of Passive and Active Measurement Workshop, Fort Collins, CO (2002)

A Simple and Fast Algorithm for Bluetooth Network Formation

Yuh-Jzer Joung[1] and Geng-Dian Hwang[2]

[1] Dept. of Information Management
joung@ntu.edu.tw
[2] Dept. of Electrical Engineering,
National Taiwan University, Taipei, Taiwan
view@iis.sinica.edu.tw

1 Introduction

We present a simple and fast distributed algorithm for Bluetooth network formation. Our algorithm requires no knowledge on the underlying physical topology, and proceeds in only one phase to connect the devices. If the physical topology is weakly connected, then the algorithm guarantees that the resulting network will be connected with high probability. The algorithm is also symmetric in the sense that each device executes exactly the same algorithm, thereby increasing the flexibility of network deployment. In contrast, previous results (e.g., [6, 3, 2, 5,10,1, 8, 9, 4]) on Bluetooth network formation require the underlying physical topology to be a complete graph, nodes to be asymmetric, or proceed in more than one phase to connect disconnect components constructed in the first phase.

2 The BlueTag Algorithm

A distinguishing characteristic of Bluetooth link formation procedure is that two devices must be in a pair of complementary states—*INQUIRY* and *INQUIRY SCAN*. The two devices hop through the same, predetermined, frequency sequence, but the one in INQUIRY state hops at a faster rate than the other so that they have a chance to 'meet' in the same frequency, and therefore to discover each other and exchange the necessary information for synchronization.

Existing algorithms either let Bluetooth devices randomly alternate between the two complementary states, or predetermine devices' states for them to stay throughout the discovery process. Our approach is to let each node alternate between INQUIRY and INQUIRY_SCAN in some pattern determined uniquely by its *root* value, initialized to its device ID *BD_ADDR*. We refer to these patterns as *state alternation sequences*. Nodes with the same *root* value have exactly the same, and *synchronized*, sequence; while nodes with different *root* values have different sequences. This design allows two nodes with different *root* values to be in complementary states, and therefore have a chance to discover each other

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1413–1417, 2005.

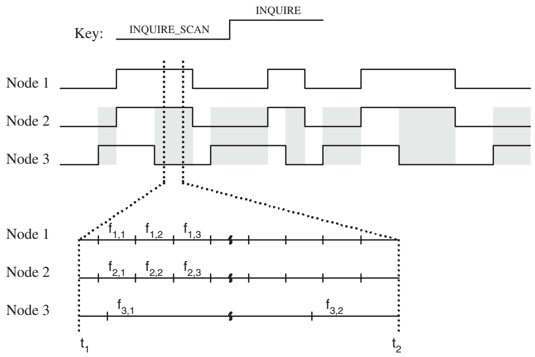


Fig. 1. State alternation sequences. The expanded interval shows the hopping frequencies

via the baseband. On the other hand, nodes with the same *root* value are always either all in state INQUIRY, or all in INQUIRY_SCAN, and so have no chance to discover one another. For example, in Figure 1, nodes 2 and 3 are in different states in the gray intervals, while nodes 1 and 2 are always in the same state because they have the same synchronized sequences.

The algorithm is outlined in Figure 2. We refer to the algorithm as BlueTag. Lines 3-7 initialize variables. In line 6, the initial state is INQUIRY; but this can be set arbitrarily. The timer *ALT_TIMER* is used to determine when to alternate state (INQUIRY or INQUIRY SCAN). After initialization, every node alternates between INQUIRY and INQUIRY SCAN according to its state alternation sequence determined by its current *root* value. When *ALT_TIMER* is timed out (lines 9-12), u computes the time to stay in the new state and then alternates to the state. State alternation sequences can be implemented by some pseudorandom number generators using a device's *BD_ADDR* as seed.

Meanwhile, if some neighbor v is discovered by u (line 13), they establish a communication link and exchange information for adjusting their state alternation sequences (line 15). Then, the one with smaller *root* value adjusts its *ALT_TIMER* and *state* to synchronize with the other. Both nodes reset their *DC_TIMER* to discover other nodes with different *root* values. We can say that a temporary piconet is formed by u and v to exchange information for synchronization (other than to know the existence of each other). However, the piconet is broken after the procedure. The two nodes then continue on their own to discover other nodes.

DC_TIMEOUT can be determined by analyzing the connection establishment time T_c between two Bluetooth devices with different state alternation sequences, and then use the following empirical formula [7]:

$$DC\_TIMEOUT = E[T_c] + \sqrt{Var[T_c]} + r_{\max} \qquad (1)$$

A node's *DC_TIMER* will time out when it cannot find any neighboring node with a different *root* value. Then it terminates its action in the network formation

```
Main(Bluetooth node $u$)
{
    $root \leftarrow u.BD\_ADDR$
    reset $DC\_TIMER$
    $neighbors \leftarrow \emptyset$
    $state \leftarrow$ INQUIRY
    $ALT\_TIMER \leftarrow -1$
    /* beginning of neighbor discovery process */
    while $DC\_TIMER$ not timed out {
        if $ALT\_TIMER$ timed out {
            compute new $ALT\_TIMER$
            $state \leftarrow \overline{state}$ /* alternate state */
        }
        if a Bluetooth node $v$ is discovered {
            $neighbors \leftarrow neighbors \bigcup \{v\}$
            $EXCHANGE\_INFO(v)$
            if $v.root > u.root$ {
                $u.root \leftarrow v.root$
                /* synchronize with $v$ */
                compute new $ALT\_TIMER$
                $state \leftarrow \overline{state}$ /* change to $v$'s state */
                }
            reset $DC\_TIMER$
        }
    }
    $ROLE\_ASSIGNMENT\ (u, neighbors)$
}
```

Fig. 2. Outline of the algorithm executed by a node u

process. Note that at this point although the overall network construction may not yet complete, the terminating node has full knowledge on its neighboring nodes in the network to be constructed. So the node may now start its upper-level application. If in the application the node needs to communicate to its neighbors and they have not yet started the application, the communicating messages can be queued until they are ready. So a node will not necessarily be blocked from starting the application even if other nodes may still be in the network formation process.

To illustrate the algorithm, consider Figure 3. We label each node with $i[j]$, where i is the node's *BD_ADDR*, and j is its *root* value. Part (a) shows the initial physical topology, where dashed lines represent physical links. In (b), node 3 and 1 discover each other and establish a link (solid line). Since node 3 has larger *root* value, it beats node 1, and makes node 1 to carry its *root* value, so as to synchronize with its state alternation sequence. In (c), 4 and 5 discover each other and establish a link. At the same time, 2 and 3 discover each other and establish a link. Moreover, node 1 terminates the algorithm (represented in gray color) because it cannot discover any node. In (d), 5 and 3 discover each

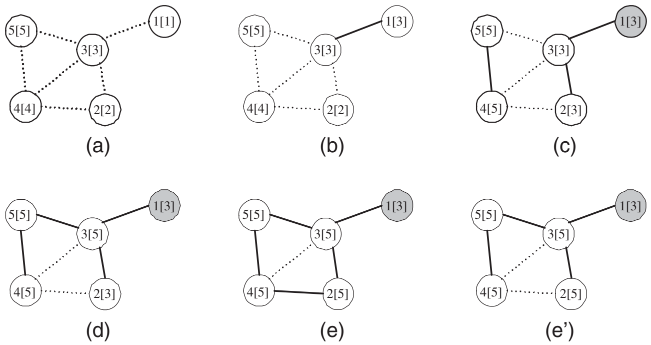


Fig. 3. A scenario of BlueTag

other and establish a link. Node 3 (with new *root* 5) now synchronizes with node 5, and so its synchronization with node 2 (still with *root* 3) is broken. Note that the previously established link between 2 and 3 (stored in their variables *neighbors*) is not affected even though each node re-synchronizes with another. In (e), 4 and 2 discover each other and establish a link. Node 2 now carries *root* value 5. So nodes 2 and 3 are now synchronized again (by *root* value 5 this time). A variation of scenario (e) (shown in (e′)) is that, instead of 4 and 2 discovering each other, 3 and 2 re-discover each other as they carry different *root* values after (d). If this is the case, then no new link will be established, as 3 and 2 already knew each other from scenario (c). However, node 2 will adjust its *root* to 5, avoiding it from re-discovering 3, or discovering 4. In either case, nodes $2, 3, 4$ and 5 will all terminate because they cannot discover any more node. From the example we can also see that the nodes may not terminate with the same *root* value (but the connected property of the final network is guaranteed).

To complete the network formation, each node needs to know which piconets it is involved in and its roles in them. This can be done by either *root-based*—x is the master of y if x has larger root value than y has when they last discover each other; *ID-based*—x is the master of y if x has larger *BD_ADDR* than y has; or *Random*—x is the master of y if x is in INQUIRY state when they last discover each other.

References

1. S. Basagni and C. Petrioli. Multihop scatternet formation for bluetooth networks. VTC Spring 2002.
2. C. Law, A. K. Mehta, and K.-Y. Siu. Performance of a new bluetooth scatternet formation protocol. ACM MobiHoc 2001.
3. C. Law and K.-Y. Siu. A bluetooth scatternet formation algorithm. IEEE Symposium on Ad Hoc Wireless Networks.
4. C. Petrioli and S. Basagni. Degree-constrained multihop scatternet formation for bluetooth networks. IEEE Globecom 2002.

5. L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal. Clustering algorithms for wireless ad hoc networks. 4th international workshop on Discrete algorithms and methods for mobile computing and communications, 2000.
6. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of bluetooth personal area networks. INFOCOM 2001.
7. T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Proximity awareness and ad hoc network establishment in bluetooth. Technical Research Report CSHCN 2001-1 (ISR TR 2001-10), University of Maryland, 2001.
8. I. Stojmenović. Dominating set based bluetooth scatternet formation with localized maintenance. IPTPS 2002.
9. Z. Wang, R. J. Thomas, and Z. Haas. Bluenet – a new scatternet formation scheme. HICSS-35, 2002.
10. G. V. Záruba, S. Basagni, and I. Chlamtac. Bluetrees-scatternet formation to enable bluetooth-based ad hoc networks. ICC 2001.

A Node Cooperative ARQ Scheme for Wireless Ad-Hoc Networks

Mehrdad Dianati, Xinhua Ling, Sagar Naik, and Xuemin Shen

Department of Electrical and Computer Engineering,
University of Waterloo,
Waterloo, Ontario, Canada N2L 3G1

Abstract. We propose a Node Cooperative Automatic Repeat reQuest (ARQ) scheme for wireless ad-hoc networks to combat the effects of channel fading on the performance of link layer retransmission scheme. The proposed scheme is inspired by the idea of *cooperative diversity*. Simulation results are given to demonstrate the significant throughput gain, even when the quality of channel between the communicating nodes is poor.

1 Introduction

It is well known that the average bit error rate over wireless channels is several orders of magnitude higher than that in wired or fiber optic channels. Automatic Repeat reQuest schemes (ARQ) are *de facto* parts of wireless link layer to combat channel unreliability and to avoid expensive retransmissions by the transport layer's error control mechanism [1][2]. The conventional ARQ schemes have been developed for wireline networks, where frame errors are random, i.e., there is no correlation between frame error probabilities for different frames. However, due to the inherent characteristics of the fading process in wireless channels, frame errors appear in bursts rather than randomly. When the link between two communicating nodes is experiencing frame errors, there is a high probability that the bad channel condition will persist for a period as long as the transmission time of multiple data frames. Conventional retransmission schemes are not very effective in such environments with bursty frame errors, and will cause significant degradation in the performance of the link layer. This is because the sender retransmits data frames by remaining oblivious to the fact that even the retransmitted packets will encounter bit errors with very high probability due to the persistence of the bad channel condition.

In this paper, we propose a simple, but efficient, ARQ scheme, namely the Node Cooperative Stop and Wait (NCSW) scheme. The proposed scheme is inspired by the idea of cooperative diversity to improve the performance of link layer protocols in wireless networks [3]. We compare the performance of the proposed NCSW scheme with the conventional retransmission scheme. Our simulation results show that cooperation of a small number of nodes can produce significant throughput gain, even when the average quality of the channel between the sender and the receiver nodes is poor.

The rest of this paper is organized as follows. The system model is specified in Section 2. The proposed ARQ scheme is explained in Section 3, and the simulation results are given in Section 4. Finally some concluding remarks are given in Section 5.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1418–1421, 2005.

2 System Model

We consider an ad-hoc wireless network model as shown in Fig. 1. For the sake of generality, a group of autonomous nodes without any central control are assumed. A single node in this model can be viewed as a mobile device, a base station or an access point. A cooperation group is a subset of nodes that can reach one another with a single hop. In other words, nodes in a cooperation group are in the radio coverage area of one another. Those groups may be set up during connection stage or link level handshaking (e.g., RTS/CTS in 802.11). A node may join several cooperation groups depending on its position, capability, and willingness to cooperate. As shown in Fig. 2, each of those cooperation groups can be modelled as a single hop wireless network. At any instant of time, one sender node captures the shared medium to send a burst of frames to its intended destination node. During that time period other nodes (i.e., neighbor nodes) in the group keep listening to the shared channel, and assist the sender and the receiver nodes if error happens.

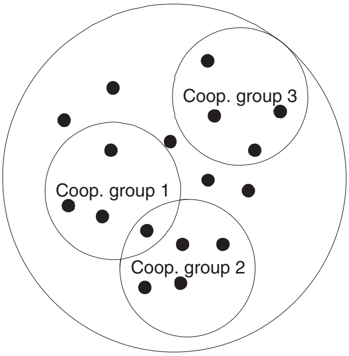


Fig. 1. An ad-hoc wireless network model

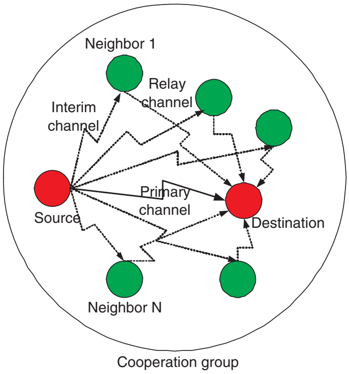


Fig. 2. A single cooperation group

Since long range wireless networks, such as satellite networks, are not considered in this paper, the propagation delays among communicating nodes are assumed to be relatively small (e.g., wireless LANs and cellular networks). In those systems, a Stop and Wait (SW) retransmission discipline is more appropriate than the Go Back N (GBN) and Selective Repeat (SR) schemes. In the SW scheme, the sender node does not transmit the next frame until the correct reception of the previous frame is confirmed by an explicit or implicit ACK. We assume that the reverse channel, used for ACK/NAK, is error-free. Thus the ACK/NAK frames can be received immediately and correctly by all the nodes in a cooperation group.

3 Node Cooperative SW Scheme

Depending on the relative velocity of communicating nodes, the duration of channel fading can be as long as the transmission time of several frames. For example, in a slow fading channel with a node speed of 5 Km/h and carrier frequency $f_c = 2400MHz$, the average fading duration is about 60 ms [4]. For a typical frame length of 5 ms, the conventional SW scheme has to retransmit the erroneous frame for an average of 12 times. The situation will be worse in high-rate systems with a shorter frame duration.

With the proposed NCSW ARQ scheme, the neighbor nodes in a cooperation group (Fig. 2) monitor the ongoing communication between a sender and a receiver node, decode the received frame, and store a copy of the most recently receicved frame. If the receiver can not decode a frame correctly, it will send a NAK to the sender. Consequently, the sender will respond by retransmitting the frame. In the SW scheme, the neighbor nodes are oblivious to the retransmissions. However, in the NCSW scheme, when the neighbor nodes in a cooperation group receive a NAK, they will transmit the requested frame to the receiver concurrently with the retransmission trials by the sender. When a frame is acknowledged by the receiver, all the nodes in the cooperation group drop their corresponding copy of the acknowledged frame. Obviously, a neighbor node in the cooperation group can retransmit only if it has already received a correct copy of the requested frame. Even if a neighbor node has a correct copy, cooperation can be optional. This guarantees the backward compatibility of the NCSW protocol with the conventional SW protocol.

4 Simulation Results

We simulate a single hop ad-hoc network with one pair of sender-receiver nodes and varying number of neighbor nodes, as shown in Fig. 2. The channels among the nodes are generated by the Rayleigh fading model. The carrier frequency is 2400 MHz and the relative speed of the nodes is 5 Km/h. The quality of the channels are represented in terms of the ratio of the fading margin and the mean value of the fading envelope, namely L. When the value of fading envelope is below the fading margin, the transmitted frame can not be decoded properly. As the value of the fading margin increases, poorer channel qualities and more frame errors are encountered. The data frame duration is assumed to be 5 ms, which is a reasonable value for many wireless data networks. Perfect ACK/NAK information on the feedback channels is assumed to be available for the whole cooperation group right after a frame transmission. The numerical results are obtained from Monte-Carlo simulations.

To observe the impact of a small number of cooperative nodes on the system throughput, the SW and the NCSW schemes with only 2 neighbor nodes are simulated. The throughput of the both schemes are obtained by simulations. As shown in Fig. 3, the results are plotted against variations of the quality of the primary channel, as denoted by L_p (L of the primary channel). The quality of all the interim and the relay channels are assumed to be identical, and denoted by L_r (L of the relay channel). To demonstrate the impact of the variations in the quality of the interim and relay channels, throughput of the NCSW protocol is plotted for two different values of L_r, namely $-5dB$ and $-1dB$, respectively. The results in Fig. 3 show that with cooperation of only 2 neighbor nodes, throughput of the NCSW scheme can be improved by up to 30%, depending on the quality of the interim and relay channels.

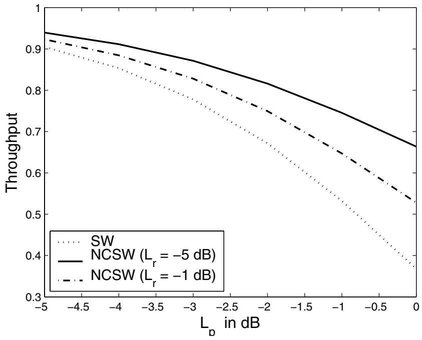


Fig. 3. Throughput vs. the fading margin

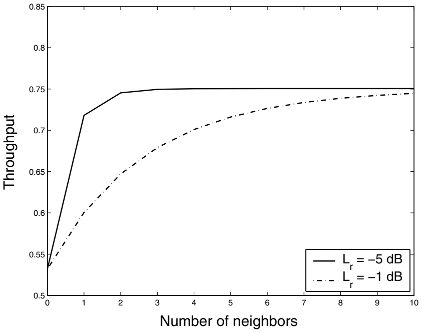


Fig. 4. Throughput vs. the number of neighbors

To observe the impact of the number of neighbor nodes on the protocol throughput, the fading margin of the primary channel is set to $L_p = -1dB$. The simulations are performed for two different fading margins for the relay and interim channels, namely $L_r = -5dB$ and $L_r = -1dB$. As shown in Fig. 4, when the number of the cooperative nodes is increased, the system throughput approaches to a saturation level depending on the quality of the primary and the interim and relay channels. If the qualities of the interim and relay channels are good, having even one or two neighbor nodes can significantly improve the throughput. However, when the qualities of the interim and relay channels are poor, more neighbor nodes are required to achieve the same level of performance gain. Saturation in the system throughput is also expected. In fact, regardless of the number of neighbor nodes or their channel qualities, individual frame errors can not be prevented. However, cooperation of the neighbor nodes can reduce the impact of bursty erros.

5 Conclusions

In this paper, we have proposed a Node Cooperative Stop and Wait (NCSW) ARQ scheme for the wireless ad hoc networks. The simulation results demonstrate that the effectiveness of retransmission scheme, in terms of throughput, can be improved significantly by cooperation among the adjacent nodes in wireless ad-hoc networks.

References

1. O. Gurbuz and E. Ayanoglu, "A transparent ARQ scheme for broadband wireless access," *Proc. IEEE WCNC'04, 2004*, vol. 1, March 2004, pp. 423 - 429.
2. A.R. Parsad, Y. Shinohara, and K. Seki, "Performance of hybrid ARQ for IP packet transmission on fading channel," *IEEE Trans. Veh. Technol.*, vol. 48, no. 3, May 1999, pp. 900-910.
3. A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity, Part I: System description" *IEEE Trans. Commun.*, vol. 51, no. 11, Nov. 2003, pp. 1927-1938.
4. G.L. Stuber, "Principles of Mobile Communication,", Kluwer Academic Publishers, 2001.

SWATCH: A StepWise AdapTive Clustering Hierarchy in Wireless Sensor Networks

Quanhong Wang[1], Kenan Xu [1], Hossam Hassanein[2], and Glen Takahara [3]

[1] Department of Electrical and Computer Engineering, Queen's University, Canada
{1qw, 1kx}@qlink.queensu.ca
[2] School of Computing, Queen's University, Canada
hossam @cs.queensu.ca
[3] Department of Mathematics and Statistics, Queen's University, Canada
takahara@mast.queensu.ca

Abstract. Distributed clustering techniques are considered effective and practical for power-saving in Wireless Sensor Networks (WSNs). In this paper, we propose a novel cluster formation scheme in WSNs, called StepWise AdapTive Clustering Hierarchy (SWATCH). SWATCH aims to solve the ClusterHead (CH) number uncertainty problem common to existing distributed clustering schemes, while keeping the desirable properties of self-organization, simplicity and dynamic adaptation. We develop a two-tier hierarchical Markov chain model to track the operation of SWATCH. Based on this model, we derive formulations of the statistical properties of our proposed scheme. Numerical results verify the design objectives in that the number of selected clusterheads highly conforms to the optimal value.

1 Introduction

Communication cost has been shown to be the major source of energy dissipation in Wireless Sensor Networks (WSNs) [1]. Clustering is an effective framework under which different energy saving techniques, such as energy-saving MAC and local data processing, can be easily adopted. With a clustered structure, the entire network is partitioned into disjoint clusters. Each cluster consists of one ClusterHead (CH) and multiple Member Nodes (MNs). Forming the virtual backbone of the network, the CHs collect data from MNs and relay the processed data, e.g., aggregated data [2] to the base station.

Although the optimal number of CHs has been proven to exist for WSNs using certain clustering scheme [2,3], clustering schemes fail to guarantee the optimal number of CHs that will be elected in a dynamical basis. For example, LEACH [2] achieves energy saving by having the network dynamically partitioned into multiple clusters and local data processing in each of them. However, due to the randomness in the CH selection algorithm, the number of CHs is not guaranteed to be equal to the expected optimal value (according to our investigation, the percentage of time that the number of CHs is equal to the optimal value is observed to be less than 20%, see Fig. 1.) In the worse case, when one CH serves the whole network, its energy would drain rapidly, and fairness is severely affected.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1422 – 1425, 2005.

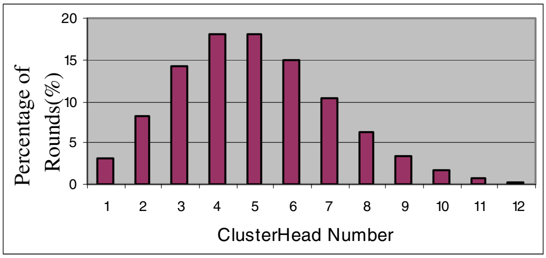


Fig. 1. CH Number Distribution of LEACH (100 nodes, Target CH Number =5. The percentage of CH number greater than 12 is negligible)

In this paper, we propose a novel single hop clustering scheme called StepWise AdapTive Clustering Hierarchy (SWATCH). SWATCH relieves the CH number variability problem by employing stepwise CH selection in two stages. It is a dynamic and straightforward scheme as LEACH. However, instead of selecting all CHs in one step, SWATCH splits the selection phase into an initial selection stage and an add-on selection stage. The initial selection is similar to LEACH. However, if the number of CHs in the initial selection is below a pre-determined target, the add-on selection will be invoked and will continue until an acceptable number of CHs have been selected. As a result, the number of CHs selected in each round tends to congregate in a narrow range around the optimal value. In other words, the variance of the number of CHs in each round is decreased. In order to evaluate the performance of SWATCH, we develop a hierarchical Markov chain model to track the behavior of the system. Numerical results verify our design objective in that the number of selected CHs highly conforms to the optimal value. Based on the results in [2,3], which show that an optimal number of CHs can greatly reduce the communication energy, we argue that SWATCH is energy efficient and fair.

The remainder of this paper is organized as follows. In section 2, the SWATCH scheme is presented. In section 3, the performance of SWATCH is evaluated by comparing to LEACH scheme.

2 SWATCH Description

2.1 Preliminaries

We make the following assumptions about the network and communication models. We assume a static network consisting of N identical stationary nodes, which are distributed randomly in a 2-dimensional sensing field (to fulfill a constant sensing task). The data is pumped out proactively at each node. A fixed base station is located far from the network to gather the collective information.

We assume that all nodes are capable of adjusting their transmission power according to transmission distance, and they can transmit with enough power to reach the BS and other nodes in the network if necessary. In order to conserve energy, each member node sends its data to its associated CH, and the CH processes the data and sends the aggregated information to the base station. The CH also performs local coordination in the cluster as in [2,3]. Moreover we assume error-free communication as in [3].

2.2 SWATCH Scheme

In SWATCH, CH selection and cluster formation is on round basis. The CH Selection Period (CHSP) is split into two stages, namely, the CH Initial Selection Stage (CHISS) and the CH Add-on Selection Stage (CHASS). In the CHISS, each CH candidate independently makes a decision to be a CH using the probability p_{Init} ($p_{Init}=(P_{Init}/i)*N)$, where N is the total number of nodes in the system, P_{Init} is the initial selection probability in a round, and i is the number of CH candidates at the beginning of the round). If the number of CHs in the CHISS reaches or exceeds a predetermined target value T_A , the CHSP terminates. Otherwise, the CHASS will be invoked. In the CHASS, all remaining CH candidates take part in the CH add-on procedure and make a decision to be CH based on the probability p_{Add_k} ($p_{Add_k} = (P_{Add} / k) * N$), where P_{Add} is the add-on selection probability and k is the number of CH candidates at beginning of the CHASS). Add-on selection is invoked repeatedly until the total number of CHs in the current round is equal to or greater than T_A. A timeline showing the rounds and stages is depicted in Fig. 2.

In order to speed up the CH selection procedure and avoid unnecessary CH selection due to a small number of CH candidates, we introduce the CHSP-End threshold $T_E(T_E>T_A)$. When the number of CH candidates is less than or equal to T_E, all CH candidates will become CHs automatically; otherwise, the normal CHISS is invoked.

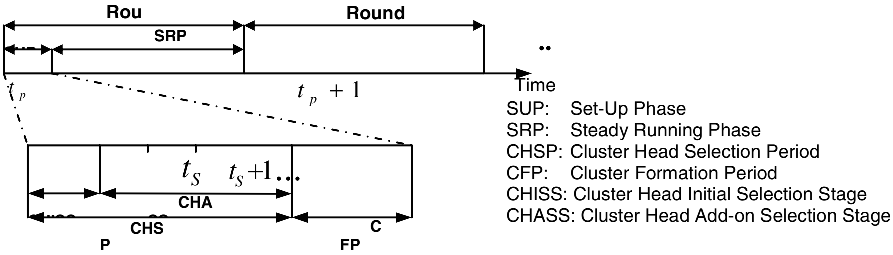

Rou
Round
SRP
••
Time
t_p
$t_p + 1$
t_S
t_S+1...
CHA
CHS
P
C
FP
SUP: Set-Up Phase
SRP: Steady Running Phase
CHSP: Cluster Head Selection Period
CFP: Cluster Formation Period
CHISS: Cluster Head Initial Selection Stage
CHASS: Cluster Head Add-on Selection Stage

Fig. 2. Illustration of SWATCH Timeline

3 Performance Evaluation

To evaluate the effectiveness of SWATCH, we develop a hierarchical Markov chain model, consisting of a Primary Markov Chain (PMC) and a Secondary Markov Chain (SMC). The PMC is used to describe the CH selection procedure from round to round, while the SMC is used to describe the CH selection behavior during the CHASS. The model is omitted due to the space limitation. Based on it, we calculate the expectation and Coefficient of Variance (C.O.V) of the number of CHs, as well as the average extra number of CH selection steps.

Fig. 3 illustrates the results for the C.O.V of the number of CHs using both LEACH and SWATCH for the same WSN with different target CH numbers. In all the cases, SWATCH produces much smaller C.O.V. than LEACH. It implies that SWATCH can ensure a more stable number of CHs.

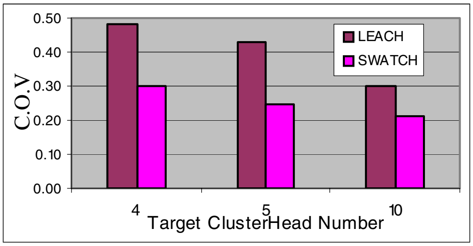


Fig. 3. C.O.V of ClusterHead Number (100 nodes, target values = 4, 5, 10)

SWATCH addresses the CH number uncertainty problem explicitly by exploiting a two-stage CH selection algorithm. Since it has been earlier shown that there exists an optimal value for individual WSNs, which leads to the minimum energy dissipation, we argue that SWATCH will provide higher energy efficiency than the existing distributed schemes.

References

1. V. Raghunathan, C. Schurgers, S. Park and Mani B. Srivastava, “Energy-aware Wireless Microsensor Networks”, IEEE Signal Processing Magazine, Vol. 19, Issue 2, pp. 40-50, March 2002.
2. W. B. Heinzelman, A. P. Chandrakasan, H. Balakrishnan, “An application-specific protocol architecture for Wireless Microsensor Networks”, IEEE Tran. On Wireless Communications, Vol. 1, No. 4, pp.660-670, Oct. 2002.
3. S. Bandyopadhyay and E. J. Coyle, “An energy efficient hierarchical clustering algorithm for Wireless Sensor Networks”, IEEE INFOCOM 2003, USA, Mar. – Apr. 2003.

Design and Performance Evaluation of WDM/TDMA-Based MAC Protocol in AWG-Based WDM-PON

Kyeong-Eun Han[1], Yang He[1], Seung-Hyun Lee[1]
Biswanath Mukherjee[2], and Young-Chon Kim[1]

[1] Chonbuk National University, Jeonju 561-756, Korea
{kehan, yckim}@chonbuk.ac.kr
[2] University of California. Davis, CA 95616, USA

Abstract. In this paper, we propose a MAC protocol for a passive optical network (PON) employing wavelength-division multiplexing (WDM) with arrayed-waveguide gratings (AWGs) performing routing functionality at the remote node. In proposed MAC protocol, we employ a Request/Permit based MAC protocol using multipoint control protocol (MPCP). It is simulated using OPNET and their performances are evaluated in terms of bandwidth utilization and queuing delay.

1 Introduction

A PON is a passive broadcast optical network consisting of an optical line terminal (OLT) at the central office connected to many optical network units (ONUs), through a passive optical device (for distribution/collection) located at a remote node (RN). Although a PON provides the necessary improvements, simplicity, scalability and low cost of implementation, they do not provide a scalable solution for access networks as the number of users is limited by the splitter attenuation. The AWG-based WDM-PON is the new technology to solve these problems. Network costs and complexity can be significantly reduced by using arrayed-waveguide gratings (AWGs) which support the wavelength-selective routing function [1]. Thus, these networks are intrinsically transparent to the channel capacity, do not suffer power splitting losses and enhance the reliability and privacy. In order to provide efficient resource allocation and use ethernet frame for carrying IP traffic more easily in WDM-PON, a appropriate MAC protocol is required. Therefore, in this paper, we propose a MAC protocol employing a Request/Permit based on MPCP for dynamic bandwidth allocation (DBA).

2 AWG based WDM-PON Architecture

The used WDM-PON architecture is shown in Figure 1. Downstream traffic is transmitted from OLT through two stages of AWGs: Stage-1 and Stage-2. For

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1426–1429, 2005.

upstream traffic we employ a separate fiber wherein the WDM signals (packets) from Stage-2 AWGs are multiplexed using TDMA. At each cluster of upstream, a given ONU (say *nth* ONU) transmits the data on a wavelength which is being used to receive downstream data in its immediately preceding ONU (say $(n-1)th$ ONU). For downstream transmission, 32 wavelengths from any input port are demultiplexed into 8 output ports by using free spectral range (FSR) property. Thus each output port in Stage-1 has 4 wavelengths from an input port. In all there are 16 wavelengths from 4 input ports which are all even or odd indices of wavelengths. As mentioned earlier, combiner at Stage-2 combines 32 ONUs with different wavelengths and 32 wavelengths (combined at Stage-2) are recombined by the combiner at Stage-1. This results in the collision in this part. Therefore, to avoid the collision and provide an efficient bandwidth allocation among ONUs sharing with same wavelength, an appropriate MAC protocol is required.

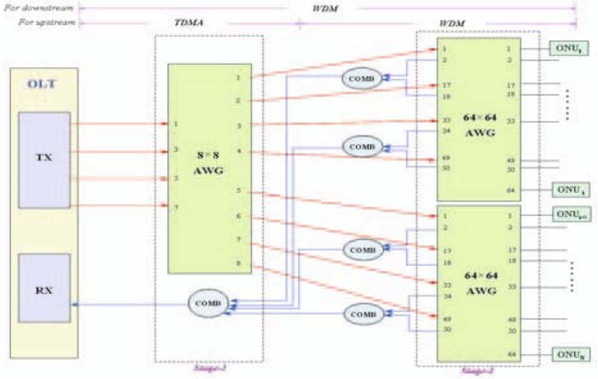
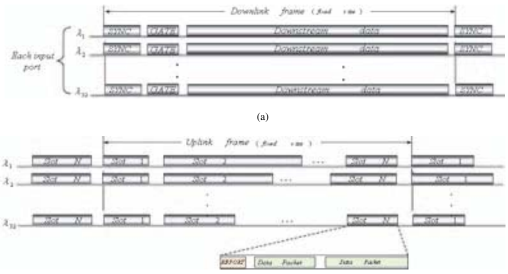


Fig. 1. WDM PON system architecture

3 Proposed MAC Protocol

In this section, we describe our MAC protocol, which controls the access of shared wavelengths for upstream transmission as well as downstream one for controlling upstream. In particular a Request/Permit based MAC protocol employing MPCP is used. We also use ethernet container for control and data packets. Because using one, the network can easily carry IP packets and can use ubiquitous/cheap hardware with the desired scalability. In MPCP the procedure for DBA relies on two control messages: GATE and REPORT. GATE is used by OLT on downstream to grant the bandwidth to ONUs for both discovery and normal transmission. REPORT is used by ONUs to inform the ONU's queue information to the OLT. GATE and REPORT are transmitted for every frame periodically. Fig. 2 (a) and (b) shows the down/upstream frame format respectively.

Fig. 3 shows the exchange procedure of control message for registration and DBA respectively. In fig. 3-(a) first, the OLT broadcasts GATE containing the

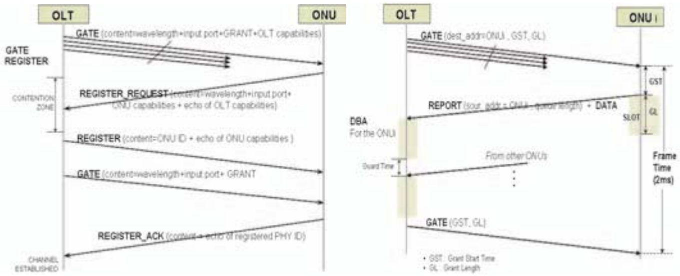


(a)

(b)

Fig. 2. (a)Downstream and (b)Upstream frame format

information of wavelength, input port number and allocated bandwidth for registration. Upon receiving GATE, active ONUs reply to OLT with the REGISTER-REQEUST. If this message is successfully received, the OLT searches the ONU-id from table and allocates it through the REGISTER. Finally, the OLT and ONU confirm to complete this procedure as exchanging GATE and REGISTER-ACK. In fig. 3(b) the OLT broadcasts GATE to all ONUs based on the mapping table. Each ONU (received the GATE from OLT) send the REPORT along with their own data to be allocated from OLT over a specific channel. After receiving the REPORT from all ONUs, OLT schedules based on requested bandwidth for each ONU sharing with the same channel and transmits the allocation results through GATE.


OLT
ONU
GATE (content=wavelength+input port+GRANT+OLT capabilities)
GATE REGISTER
CONTENTION ZONE
REGISTER_REQUEST (content=wavelength+input port+ ONU capabilities + echo of OLT capabilities)
REGISTER (content=ONU ID + echo of ONU capabilities)
GATE (content=wavelength+input port+ GRANT
REGISTER_ACK (content = echo of registered PHY ID)
CHANNEL ESTABLISHED
OLT
ONU i
GATE (dest_addr=ONUi , GST, GL)
GST
REPORT (sour_addr = ONUi , queue length) + DATA
SLOT
GL
DBA For the ONUi
Guard Time
From other ONUs
Frame Time (2ms)
GATE (GST, GL)
• GST : Grant Start Time
• GL : Grant Length

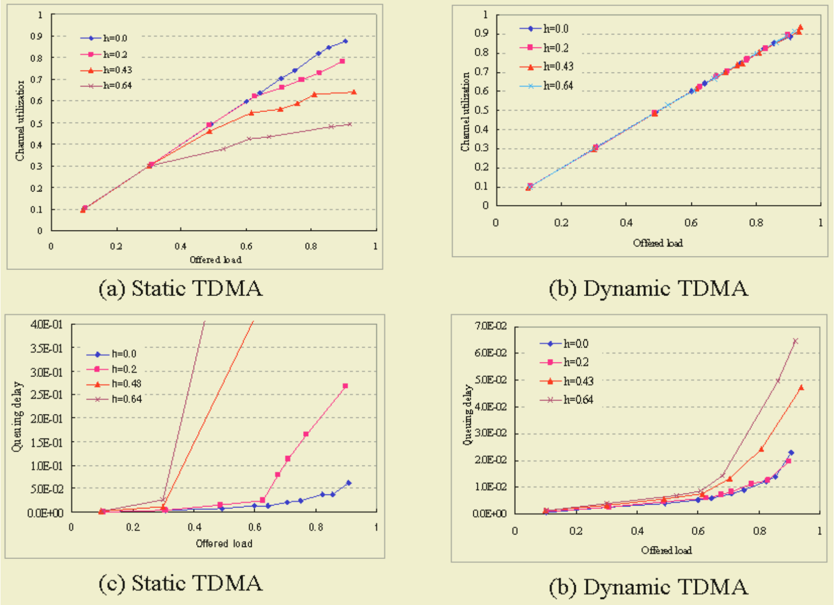
(a)

(b)

Fig. 3. Procedure of (a) registration and (b) dynamic bandwidth allocation

4 Performance Evaluation

We simulate and evaluate the performance of proposed MAC protocol using OPNET under the hot spot ratio (h). We assume 128 ONUs, 32 wavelengths, 96 bits inter-packet gap (IPG) and $1\mu s$ guard time for simulation. We also assume the fixed frame size ($2ms$) and 1Gbps-channel capacity. Fig. 4 present the plots of utilization and queuing delay vs. offered load for the static and dynamic bandwidth allocation.



(a) Static TDMA

(b) Dynamic TDMA

(c) Static TDMA

(b) Dynamic TDMA

Fig. 4. Channel utilization and queuing delay both MAC protocols

5 Conclusion

We proposed the MAC protocol based on a Request/Permit for DBA. It employs a minimum bandwidth guaranteed algorithm. The simulation results indicated the proposed MAC protocol can significantly improve the bandwidth utilization and queuing delay regardless of hot spot ratio.

Acknowledgement

This work is supported by IITA and Korea Industrial Technology Foundation(KOTEF), Republic of Korea.

References

1. Guido Mario, "Design and Cost Performance of the Multistage WDM-PON Access Networks," J. Lightwave Technology, vol.18, pp.125-143, February 2000.

A Backup Tree Algorithm for Multicast Overlay Networks

Torsten Braun[1], Vijay Arya[2], and Thierry Turletti[2]

[1] University of Bern, Neubrückstrasse 10, CH-3012 Bern
braun@iam.unibe.ch
[2] INRIA, 2004 route des Lucioles, B.P. 93, F-06902 Sophia Antipolis
{Vijay.Arya, Thierry.Turletti}@sophia.inria.fr

Abstract. We propose the so-called backup tree algorithm to compute a set of n-1 backup multicast delivery trees from the default multicast tree for application level multicast. For each backup multicast tree exactly one link of the default multicast tree is replaced by a backup link from the set of available links. The backup tree algorithm calculates the n-1 trees with a complexity of O (m log n).

1 Introduction

Mechanisms for explicit path selection [1] are not included in most application or IP level multicast distribution concepts. We propose that a sender of a multicast packet can select a backup multicast tree instead of the default multicast tree by inserting a fixed size tree identifier into the multicast packet. This allows a sender selecting individual multicast trees for each single packet in order to react immediately on events such as link breaks, node failures, and group member leaves. Delays for reestablishing new multicast delivery trees for new topologies can be avoided. Load balancing using different trees simultaneously can be applied for increasing throughput or if particular links of the default multicast tree become congested. Explicit path selection is also useful in secure group communications for preventing particular nodes of a multicast group to receive a multicast message.

The chosen tree must be identified within each multicast message by a tree identifier. Since we assume a limited identifier space, we have to limit the set of possible trees among which a sender can choose. We propose the so-called backup tree algorithm that calculates n-1 backup multicast trees belonging to a single default multicast delivery tree for an overlay of n nodes. The algorithm also minimizes the number of backup links that are required to build the n-1 backup multicast trees. Each of the n-1 backup multicast trees has n-2 links in common with the default multicast tree and differs in exactly one link. A sender of a packet can then choose among n trees to distribute a multicast packet.

We developed our concept for application level multicast [2], because it is a promising basis for future multicast services and applications. We assume some kind of overlay network on top of which the application level multicast protocol can run. We

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1430 – 1434, 2005.

further assume that the underlying protocols establish a mesh of links between nodes, not necessarily a full mesh, and that a spanning tree for multicast data delivery is constructed from the mesh. A certain connectivity for each node is required to allow the identification of backup links that shall replace the links of the default multicast tree in certain conditions. Scalability is limited by the overhead to distribute topology information and to compute a spanning tree based on this information, but can be achieved for large groups by introducing hierarchical structures [3].

2 Modes and Signaling Requirements

The mechanism for computing backup multicast trees can be used in three modes:

1. ***Independent mode.*** Each node must get a complete view of the multicast overlay topology for independent calculation of the backup multicast trees. All multicast overlay nodes perform the identical algorithm and compute the same set of backup multicast trees. In order to get this complete overlay topology view, the exchange of topology information using some signaling protocol is required.
2. ***Distributed mode.*** The exchange of complete topology information can be avoided by a distributed version of the algorithm. It also allows a node to know only the local connectivity, but it requires a sophisticated signaling protocol.
3. ***Central mode.*** The algorithm can also be used at the root of the multicast delivery tree only. In this case, only the root calculates an appropriate tree for multicast data delivery and specifies the tree using a self-describing specification based on a cardinal representation of the multicast tree in the multicast data packet.

To support the independent mode, we propose a simple signaling protocol for complete topology information exchange within the multicast overlay network as a part of group management. Three signaling messages are used: join, leave and tree. The ***join*** message is sent by any node that wants to join the multicast group. It contains information about the connectivity of the new member to other peers and is forwarded towards the root of the multicast tree. In response to a join message, the root sends a ***tree*** message to the group members possibly after some admissioncontrol. The tree message updates connectivity information and informs the other group members about nodes that joined or left the multicast tree. After receiving the tree message each node updates its information about the overlay network. It calculates the alternative multicast delivery trees, i.e. the default multicast tree, e.g. using a minimum spanning tree algorithm, and the backup multicast trees using the backup tree algorithm. A node receiving a message can derive from the tree identifier (a hash of the cardinal representation) and the topology knowledge how to forward the message. If a peer node wants to leave a group, it sends a ***leave*** message to the root. Again, the root updates its member list as well as the overlay network topology and sends a tree message to the group. All group members must update their group membership and topology information and recalculate the alternative multicast delivery trees including their tree identifiers. Tree messages should be refreshed periodically.

3 Backup Tree Algorithm

We assume that n nodes of an overlay network (vertices) are interconnected by a mesh of m links (edges). For a full mesh with n vertices, the number of edges is m = n (n-1) / 2. However, usually application level multicast approaches do not establish full meshes, but only certain links between nodes. Due to existence of firewalls we have to assume that not each node can connect arbitrarily to any other node. On the other hand, in most approaches each node tries to establish a certain number of links to other nodes. This also improves the reliability of the overlay network. Out of the finally resulting mesh a huge number of possible multicast delivery trees could be calculated. We assume that multicast delivery trees are spanning trees consisting of n vertices and n-1 edges. Since a tree identifier is limited to a certain size, we need an algorithm that restricts the number of possible multicast delivery trees. We propose to restrict this number to n and to compute n-1 backup edges that can replace each of the n-1 edges of the default multicast tree in case of link breaks or congestion. We compute a backup edge from the set G-T (G: set of edges of the graph, T: set of default multicast tree edges) for each edge of the default multicast tree T. Replacing each of the n-1 edges of T by its corresponding backup edge results in n-1 backup multicast trees. The default multicast tree and the n-1 backup multicast trees result in n alternative multicast delivery trees that can be used for packet delivery over a multicast overlay network.

In the following we present the algorithm for the independent mode and we assume that each node knows the default multicast tree. There are several ways how to build a default multicast tree such as multicast routing protocols or minimum spanning tree algorithms. Assuming Prim's minimum [4] spanning tree algorithm with a complexity of O (m log n), one could naïvely calculate n-1 minimum spanning trees for the n-1 graphs G-e_i with e_i = edge i of the minimum spanning tree, i = 1, …, n-1. The complexity for calculating n-1 backup multicast trees is O (m n log n) in this case. Our intention is not to determine backup multicast trees with optimal link weights, but those that can be calculated at low cost. We achieve a complexity of O (m log n). Another goal is to determine a rather small set of edges that can serve as backup edges for the edges of the default multicast tree. Moreover, we select backup paths in such a way that they have minimum overlap with the default path. The idea for the backup tree algorithm is taken from the observation that a backup edge can repair all other edges of the default multicast tree with which the backup edge forms a cycle. For example, in Fig. 1 edges (C, H), (H, L), (C, E), (E, I), (I, M), and (M, N) can be repaired by edge (L, N). On the other hand, edge (S, A) can not be repaired. A broken edge of the default multicast tree can either be replaced by a backup edge that is not in the default multicast tree or it can not be repaired at all.

In the following we assume that the complete graph is stored in a linked data structure of vertices with pointers to their edges and neighbor vertices as it would result from a minimum spanning tree calculation. We also assume that the default multicast tree edges are labeled as such and that each vertex has a vertex identifier. First, we calculate the path from the root to each vertex in T and store this path with each vertex. Then, we calculate for each edge in G-T the resulting path from the root via the first vertex to the second vertex of the edge. That path is called backup path for a certain vertex, since it allows reaching a vertex via an alternative path than the default path along the default multicast tree. The backup paths can be used to reach vertices in case of edge failures. For example, edge (L, N) stores backup path SACHLN (S→L, N), while (N, L) stores backup path SACEIMNL (S→N, L). Edge (A, D) stores backup path SAD (S→A, D), while edge (D, A) stores SABDA (S→D, A), which will later be detected as invalid, because A occurs twice in it. A vertex can be reached via the path along the default multicast tree and via one or more backup paths. For example, node L can be reached via the default multicast tree, but also via two backup paths going via J or N. Moreover, we calculate and store for each edge in G-T at which node the path along the default multicast tree and the backup path to reach a particular node begin to differ. For example, node L can be reached via backup path SACEIMNL (S→N, L) and the default multicast tree path SACHL (S→L). These two paths begin to differ in the fourth vertex (E/H), but the first three vertices (SAC) are identical. Therefore, we store the value 3 (called common path length) with the backup path (SACEIMNL/3) at (N, L).

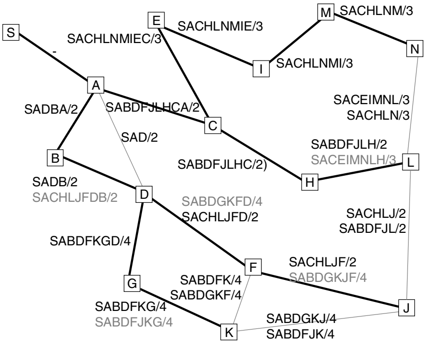


Fig. 1. Backup Multicast Tree Construction

Next, we copy the backup paths to the leaf edges of the default multicast delivery tree. We extend each copied backup path by that node of the leaf edge that is closer to the root of the tree. If a leaf edge connects to two edges of G-T, we only keep the backup path with the lowest common path length. For example, edge (H, L) connects to two edges in G-T: (N, L) and (J, L). We consider backup path SABDFJLH/2 (copied from (J, L)) as the best choice and only keep that one, but we remove SACEIMNLH/3 (copied from (L, N)). After processing all leaf edges we consider these as labeled. We have then to process all other unlabelled edges in T in the sameway as the leaf edges. After processing leaf edges in the first round, edges (D, F), (D, G), (C, H), and (I, M) are processed in the second round. Basically, the backup paths are copied towards the root of the tree. For edge (D, F) the backup path copied from edge (K, F), i.e. SABDGKFD/4, is copied, but becomes removed, because the other backup path (SACHLJFD/2) from edge (F, J) has a lower common path length (2 vs. 4). In a later round, edge (A, C) is considered. Only the backup path copied from (H, C) is appropriate (SABDFJLHCA/2). The other one copied from edge (C, E) SACHLNMIECA/3 contains C twice and must therefore be removed When edge 1 (S, A) is considered, all backup paths from edges (A, C) and (A, B) contain A twice. Therefore, (S, A) can not be repaired and is marked as not repairable. Fig. 1 shows the final result of the algorithm. A performance evaluation can be found in [5].

References

1. H. T. Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, A. Gandhi: BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet, *ACM SIGCOMM 2003 Workshop on Future Directions on Network Architectures (FDNA)*, August 25-27, 2003, Karlsruhe / Germany.
2. Y. Chu, S. Rao, S. Seshan, H. Zhang: A Case for End System Multicast*, IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 8, October 2002
3. S. Banerjee, B. Bhattacharjee, Ch. Kommareddy: Scalable Application Layer Multicast, *ACM SIGCOMM 2002*, Pittsburgh / USA, August 19-23, 2002
4. R. Prim: Shortest Connection Networks and Some Generalizations, *Bell System Technical Journal, Volume 36*, pp. 1389-1401, 1957
5. T. Braun, V. Arya, T. Turletti: Explicit Routing in Multicast Overlay Networks, *INRIA Research Report 5397*, November 2004

Tradeoffs for Web Communications Fast Analysis[1]

Olivier Paul and Jean-Etienne Kiba

GET/INT LOR Department, 9 rue Charles Fourier,
91011 Evry cedex - France
Olivier.Paul@int-evry.fr,
Jean-Etienne.Kiba@int-evry.fr

Abstract. In this paper, we explore a novel approach to perform a probabilistic fast analysis of web communications. Instead of relying on pattern matching algorithms, we look at simple network and transport level parameters and try to infer what happens at the application level. Our approach provides the ability to perform a trade-off between analysis speed and precision that could prove useful for some traffic analysis applications.

Introduction

The appearance of new threats (e.g. worms, DDOS attacks) has led network operators to provide intrusion detection services to their customers. In this paper we consider one of the challenges implied by this activity; the ability to monitor communications within operator networks. This task can be considered challenging for several reasons (limited traffic analysis abilities, large amounts of traffic, limited ability to introduce new mechanisms). We focus on HTTP based communications because they constitute one of the largest aggregate of packets on the Internet. The goal of this paper is to present techniques that would enable such communications to be analyzed in the network while complying with the aforementioned limitations.

Measurement Information

Our goal is to check whether network or transport level information could be used to infer application level operations. In a first part we examine how protocols might render this operation difficult. HTTP [1] exchanges can be viewed at several levels. At the lowest level, the HTTP protocol is based on a request-response protocol where each request attempts to perform an HTTP operation on an object at the server. We later call this level micro-session level. Information in HTTP 1.1 messages is organized into information elements called headers. Although HTTP 1.1 defines more than 40 different headers, requests and responses usually only use a few them.

HTTP 1.1 provides the ability for web clients and servers to multiplex several HTTP request-responses exchanges over a single TCP connection. Among persistent

[1] This work is funded by IST DIADEM Firewall and GET DDOS projects.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462,, pp. 1435 – 1439, 2005.

connections we can additionally distinguish between connections using pipelined requests and regular connections. Pipelined connections are used by the client to perform several requests without waiting for an answer from the server. This ability is however limited by the structure of html documents. Therefore micro-sessions can be distinguished at the network level by either looking at:

- Connections set-up and ending in the case of non persistent connections.
- Request-Response session patterns [3] in the case of non-pipelined persistent connections. These patterns can be found by observing TCP sequence numbers.
- Request-Response session patterns in the case of persistent pipelined connections. However only the first micro-session can be distinguished from other exchanges.

An interesting question is thus whether pipelined, persistent connections are supported in the real life. [3] shows that most browsers are either unable to use persistent-pipelined connections or configured by default to avoid using them.

Method and Objects Size Inference

Our assumption is that objects sizes can be inferred from network or transport level measurements. Several factors can play a role in making this process more difficult. For example at the transport level, measurement information includes HTTP headers.

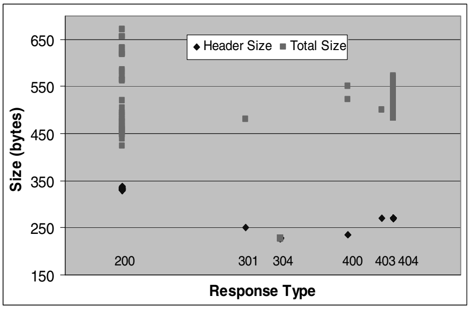


Fig. 1. Header and total sizes for various types of responses

Figure 1 provides the relation between header sizes, types of response and total sizes in the case of our web server. These values where obtained by capturing responses packets from the server over several hours. Six types of responses (identified by code numbers) were captured.

Figure 1 shows that 200 (OK) responses can be distinguished from other responses by looking at the total size. Some 200 responses have a size that collides with other types of responses. However objects associated with these responses constitute less than 1% of existing objects. 304 (not modified) responses can also be distinguished from other responses by looking at the total size.

Additionally, tests show that persistent connections include additional HTTP headers. These headers have a fixed size (57 bytes). As a result knowing whether a connection is persistent is sufficient to deduce the influence of the persistence on the HTTP header size. This knowledge can be obtained by looking at multiple connections establishment-teardown over short periods of time. **Table 1** provides the relation between response sizes and codes.

Table 1. Response code classification using response size for persistent connections

Result	**Response Size**
200	RS >550 or 250<RS<460
304	240<RS <250
301, 400, 403, 404	460<RS<550

Using a similar methodology, we define a set of classification criterion in order to infer the method used in HTTP requests.

Figure 1 shows that 200 responses can carry HTTP headers whose size are not fixed. As a result using an average HTTP header size value to estimate objects sizes in the case of GET requests can lead us erroneous results. By looking more closely at HTTP headers we can classify header fields according to their behavior:

- Some headers values never change (e.g. response code, server id, accept range).
- Some header values change but have a fixed size (e.g. last modified and date).
- Some header sizes change depending on the document (e.g. content type, size).

As a result for a given object, the response size should remain constant. This means that by keeping the relation between response sizes and object sizes, we can get an exact estimate of objects sizes.

URI Inference

For URI Inference, our goal is to use parameters such as the object size, the date and time at witch a request was performed or the IP address of the requesting client. The relation between these parameters can be found in log files on the web server.

The model we selected to perform inference operations is a Bayesian network. Bayesian networks are graphical models that can be used to represent causal relationships between variables. A Bayesian network is usually defined as an acyclic directed graph $G, G = (V, E)$, where V is a set of nodes and E the set of vertexes, a finite probability set (Ω, Z, P) and a set of variables defined on (Ω, Z, P), so that :

$$P(V_1, V_2, \dots, V_n) = \prod_{i=1}^{n} P(V_i | C(V_i)) \tag{1}$$

where $C(V_i)$, is the set of causes for V_i in the graph.

The inference in a causal network consists in propagating unquestionable information within the network, in order to deduce how beliefs concerning the other nodes are modified. This propagation is related to causal relations between nodes.

In order to limit the resources used by our model, a first step was to aggregate possible parameter values. IP addresses were aggregated into country codes and date and time were also aggregated in some way. As the cost of inference in a Bayesian network increases exponentially with the number of variables we also evaluated the ability for each parameter (size, country codes, time, date) to explain URIs. This led us to the selection of country code and size variables (Bayesian network in Figure 3).

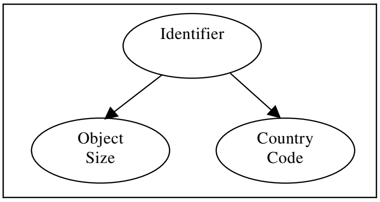


Fig. 2. Resulting Bayesian network

Implementation and Tests

A traffic analyzer was implemented as an extension to IPFilter [4] to capture HTTP sessions. Sessions are delimited as specified in section 0. When a session ends, the corresponding information (size, IP addresses, time, and date) is handed to a user space process and stored in a file.

Validation tests were performed using our departmental web server. This web server includes roughly 15k objects and receives 10k requests a day. Models were built using a one month log file including 309k requests. The validation was performed by simulating requests to the server using the same log file. Results are presented in **Table 2**. Among requests, only requests with a "200" response codes were used for method inference. Only inferred "GET", "200" requests were used for object size and URI inference.

Table 2. Ability to predict correct parameter values

Parameter	**Requests considered**	**Correct Responses**	**% Correct**
Response Code	300986	288568	96
Method	228189	208347	92
Object Size	154289	150505	98
URI	154289	135212	88

An implementation of our inference process was performed in C in order to test its performance. The implementation was performed on FreeBSD using a 2.4Ghz Pentium Xeon (512KBytes cache, 1GBytes RAM). Results show that our inference process should be able to analyze roughly 1M requests per second using a 2.5Mb model. Assuming standard Internet traffic this would allow us to treat a 20Gb/s full duplex link.

References

1. R. Fielding and al. HTTP 1.1, RFC 2616. Internet Engineering Task Force, June 1999.
2. B. Krishnamurthy and al. PRO-COW: Protocol Compliance on the Web, USITS '01, 2001.
3. F. Donelson Smith and al., What TCP/IP protocol headers can Tell Us About the Web, CM SIGMETRICS/Performance 2001, June 2001.
4. Darren Reed, IPFilter v 4.1.3, available at coombs.anu.edu.au/~avalon/, 2004.

Benefits of Multiple Battery Levels for the Lifetime of Large Wireless Sensor Networks

Mihail L.Sichitiu[1] and Rudra Dutta[2]

[1] Electrical and Computer Engineering dept.,
[2] Computer Science dept.,
Box 7534, North Carolina State University, Raleigh, NC 27695
dutta@csc.ncsu.edu

Abstract. In large wireless sensor networks, the few nodes close to the monitoring station are likely to prove the bottleneck in the useful lifetime of the network. We examine a strategy of equipping these nodes with a larger share of the total initial energy (battery) than the others, and generalizing this notion to the rest of the network. We solve a design problem involving optimizing the network lifetime using no more than a given number of distinct battery levels, and verify the results from the model by direct simulation.

1 Introduction

Wireless sensor networks have come to be recognized as an important research area in recent times [1]. In many practical sensor networs, the nodes are stationary and send periodic sensor readings to a monitoring station or sink. Thus the traffic is of an *egress* pattern and the routing is static. Under these conditions, it is well understood that the *first tier of nodes* (the nodes within direct radio range of the sink) will expend battery at the highest rate, because all traffic in the network is forwarded by this set of nodes. Similarly the second tier of nodes expend battery at the next highest rate, and so on. If all nodes possess the same initial battery level, the effective lifetime of the network is defined by the lifetime of the first tier. This problem is recognized in literature, and various strategies have been advanced to address it. For lack of space, we cannot discuss them comprehensively, see [2] for a full bibliography. In particular, it has been shown [3] that routing cannot alleviate this problem. Our current work is based on a strategy that has not been so far addresssed in literature, that of equipping different nodes with different levels of initial battery *i.e.* redistributing the total energy budget unequally among the nodes.

2 Design Problem

In this work we only consider a circular sensor field with the sink at the ceter, with roughly uniform sensor density. We assume that after deployment an arbi-

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1440–1444, 2005.

trary shortest hops routing is set up by the nodes, and is recomputed whenever a node dies. With this in mind, we introduce the following notation:

N The initial number of nodes in the network.
R_M The radius of the network *i.e.*, the distance between the sink and the node farthest from it.
n The (uniform) density of node placement over the sensing area $= \pi R_M^2/N$.
R_{TX} The transmission radius of the sensor nodes.
T The number of tiers in the network, $T \approx \frac{R_M}{R_{TX}}$.
B The total energy budget.
b_u Initial energy of each node under uniform allocation, $b_i = B/N$.
P The period between the generation of two consecutive packets by a node.
e_F The energy required to forward a packet.
e_G The energy required to generate a packet.
e_I The energy expended by each node in each period P that is independent of the data traffic.
τ The traffic generated by the nodes in a unit area, in unit time.
β The amount of energy required to transmit one unit of traffic, once.

With the assumption of a uniform density, the number of nodes in tier i is easily seen to be $N_i = (2i-1)N/T^2$. We need only consider that the flows originated in all nodes outside tier i are forwarded by the N_i nodes of tier i to obtain the time at which tier i nodes will die as: $L_i = T^2 b_u N_i P / \{N\left(T^2 - i^2\right) e_F + (2i-1)N(e_G + e_I)\}$. For any i, the above is true only if no nodes in any other tier die earlier than L_i. Thus L_1, the minimum one, defines the lifetime of the network. At this time, each node in a tier $i > 1$ will only have actually consumed b_{e_i} of its energy given by:

$$\frac{b_{e_i}}{b_u} = \frac{L_1}{L_i} = \frac{\frac{T^2-i^2}{2i-1}e_F + e_G + e_I}{(T^2-1)e_F + e_G + e_I}. \tag{1}$$

Given these preliminaries, we now define our battery reallocation problem as follows: Given a total battery budget B and a maximum of k distinct battery levels, determine the optimal battery levels $b_1, \ldots, b_k$, $b_1 > b_2 > \ldots > b_k > 0$, and their assignment for each tier of nodes in a sensor network with parameters $N, R_M, R_{TX}, P, e_F, e_G$ and e_I, such that **the total lifetime L of the network is maximized**.

The problem is trivial when $k = T$, and becomes a hard one of integer optimization when T is significantly larger. We consider the asymptotic situation when the density of nodes per unit area remains the same, but the number of nodes per unit area increases sufficiently that the density can be considered to hold for any area, however small. Here, we develop the theory only considering the forwarding energy e_F and setting $e_G = e_I = 0$; in [2] we show how to take these parameters into account. Then the energy $e(r)$ expended by the nodes in a unit area around radius r in unit time is given by $2\pi r R_{TX} e(r) = \pi(R_M^2 - r^2)\tau\beta$, hence the battery power $b(r)$ expended by each individual node in that area is given by $b(r) = e(r)/n = R_M^2 - r^2)\tau\beta/(2rR_{TX}n)$. The approximation in using the continuous model arises from the assumption that the above is valid for all values of r. In reality, this is valid for the radii $r = (2m+1)R_{TX}/2$. As a check, we plot the curve $b(r)/b_u$ as given by the above definition of $b(r)$ as well as the values of b_{e_i}/b_u as given by 1 for a network of 20 tiers, in Fig 1. As expected, the values match quite well at the middle of each tier. But the former curve is continuous, this allows us to employ derivative methods to investigate the relationship of the lifetime and the battery budget.

The energy E expended in the whole network in unit time can be easily obtained by integrating $e(r)$ over the entire area as $E = (2/3)\pi R_M^3 \tau\beta/R_{TX}$, and multiplying E by a lifetime of L seconds gives the minimum amount of total energy budget that can achieve a total lifetime of L, when distributed as $b(r)$ above, such that every node exhausts its battery at the same time. Naturally, this requires continuously varying battery levels. If we are realistically constrained to using only k distinct battery levels, then the best lifetime will be obtained by approximating this battery distribution as far as possible. The nodes in an annular area will have the same battery level, and, as before, annular areas closer to the center should have higher battery levels. Let all nodes from radius $\rho_0 (= 0)$ to ρ_1 have a battery level of b_1, from ρ_1 to ρ_2 have a battery level of b_2, and so on, until all nodes from radius ρ_{k-1} to $\rho_k (= R_M)$ have a battery level of b_k. Call these the “rings” $1 \ldots k$. The problem is to determine $\rho_1 \ldots \rho_{k-1}, b_1 \ldots b_k$. We proceed by observing that the lifetime of each ring equals the lifetime of its innermost tier (we assume rings start at tier boundaries). For ease of notation, we define the variables $r_i, i = 1 \ldots k$, as $r_i = \rho_{i-1} + R_{TX}/2$, and $r_{k+1} = R_M + R_{TX}/2$. We already have $r_1 = R_{TX}/2$. For equal lifetimes of the tiers t_i, and, hence, the rings i, the battery levels b_i must be in the same ratio as $b(r_i)$. Letting L stand for the equal lifetime thus achieved, we can assert that $b_i = b(r_i)L = (R_M^2 - r^2)\tau\beta L/(2r_i R_{TX} n)$. Since b_i sums to B over all nodes, we can obtain $B = S\pi\tau\beta L/(2R_{TX})$, where $S = \sum_{i=1}^{k} (R_M^2 - r_i^2)(r_{i+1}^2 - R_{TX} r_{i+1} - r_i^2 + R_{TX} r_i)/r_i$.

S can be minimized by setting each partial derivative of S w.r.t. r_i to zero: the particular structure of S is conducive to this procedure. This is because r_i only appears in the $i-1$-th and i-th terms of the sum S. Thus we obtain

$$\frac{\partial S}{\partial r_i} = \begin{array}{l} 2R_M^2 r_i/r_{i-1} - R_M^2 R_{TX}/r_{i-1} - 2r_{i-1}r_i \\ +R_{TX} r_{i-1} - R_M^2 r_{i+1}^2/r_i^2 + R_M^2 R_{TX} r_{i+1}/r_i^2 \\ -R_M^2 - r_{i+1}^2 + R_{TX} r_{i+1} + 3r_i^2 - 2R_{TX} r_i. \end{array} \qquad (2)$$

The above can be solved symbolically in MATLAB to obtain numerical solutions to r_i for any given problem instance. These values can in turn be used as outlined above to obtain the actual battery levels b_i to be used.

3 Simulation Results

Since existing network simulators such as OPNET focus on details of physical and MAC layers (which are not the focus of our work), they are unsuitable for

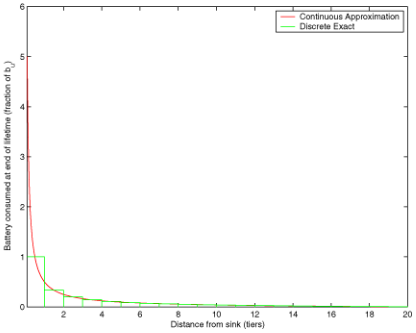


Fig. 1. Relationship of discrete battery consumption levels with continuous approximation

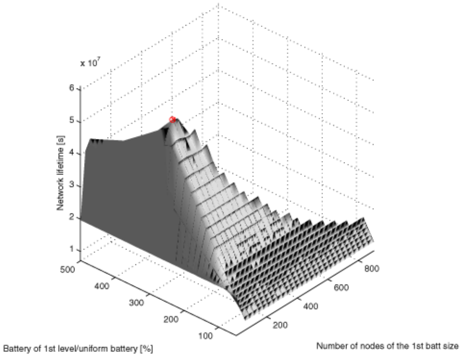


Fig. 2. Experimental validation of the best placement and best battery levels of two levels of batteries

simulation of networks of hundreds of nodes over long times. We developed a discrete even based simulation to focus on the energy consumption problem. As a base case, we considered a circular area in which 905 nodes were distributed in a square grid with the sink in the center. Each node was assumed to have power consumption characterstics and initial energy level similar to those of Berkeley motes. To assess the effect of the idealizations in Section 2 on the optimality of the battery level choices, we performed a brute-force search for the optimal values for the case when the total initial energy budget was redistributed among the nodes using two battery levels. Fig. 2 depicts the lifetime of the network as a function of the battery level of the first level of nodes and the number of nodes in the first level. The maximum lifetime is marked with the symbol '`x`'. The maximum, as predicted by theory in Section 2, is marked by the symbol '`o`'. As we can see, the location of the optimum as predicted by the continuous model matches that obtained by simulation near-perfectly.

We also simulated many deviations from the idealizations of the base case in order to verify that the approach continues to be useful under departures from the ideal. In particular, we have verified this for variations in node density, some non-uniformity in node distribution, and total number of network nodes; details can be found in [2].

In conclusion, we examined the approach of non-uniform battery allocation to the nodes of a sensor network to alleviate the problem of very early disconnection caused by the large traffic forwarding load imposed on the nodes close to the sink due to the egress nature of traffic in the network. Under a total battery budget, we demonstrated a method to approximate the optimal battery levels and number of nodes for each battery level. With this strategy, we showed that the lifetime of the network can be significantly improved, even if a small number of battery levels is used.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communication Magazine **40** (2002) 102–116
2. Sichitiu, M.L., Dutta, R.: Benefits of multiple battery levels for the lifetime of large wireless sensor networks. (CACC Technical Reports, 2005)
3. Alonso, J., Dunkels, A., Voigt, T.: Bounds on the energy consumption of routings in wireless sensor networks. In: WiOpt '04. (2004)

Enhanced Data Services in GPRS Networks via Auction-Based Prices for Admission

Saravut Yaipairoj and Fotios C. Harmantzis

Telecommunications Management, Stevens Institute of Technology,
Castle Point on the Hudson Hoboken, NJ 07030, USA
{syaipair, fharmant}@stevens.edu

Abstract. We propose an auction-based pricing scheme for admission in a GPRS network, that could enhance the GPRS data service. Important features of our approach are the partitioning of dedicated channels for enhanced data service and the employment of an auction mechanism based on revenue-maximizing auctions (optimal auctions).

1 Introduction

GPRS data service fails to support most real-time applications, because of its low data throughput. The main cause is due to the resource-sharing between voice and data traffic [3]. Since voice has priority over data, there are only few traffic channels available for data traffic. To be specific, GPRS traffic can capture approximately one or two traffic channels out of eight channels during data transmission. Performance could be even worse during the congestion time. Therefore, it is quite difficult for GPRS to support real-time data services, as long as the problem of resource sharing between voice and data remains. In [1], Odlyzko has proposed a simple but effective scheme that allows Internet users to experience better quality of service, namely, Paris Metro Pricing (PMP). He proposed partitioning of networks and pricing them differently. His technique allows users to experience a better quality of service in the higher-priced network. Even though PMP has not been implemented yet in real networks, since Internet bandwidth has become less expensive, we argue that it could be used in bandwidth-limited IP based network, such as GPRS. To enhance the GSM/GPRS service, we could partition the traffic channels and apply different prices to each group of channels. Since our focus is enhancing the data service, we argue that higher price channels should be dedicated for data service. By doing this, we could lose the efficiency of spectrum usage and degrade both voice and GPRS service in the lower price channel group. To compensate for the potential lost of revenue from channel partitioning, the dedicated channels for "enhanced" GPRS service must be limited and only allocated to users who value them most. Therefore, instead of adopting simple pricing policies for channels, we argue that auction-based prices would be more suitable [2]. In this paper, we propose auction-based pricing for admission to enhance GPRS services. The merits of our proposed model are: a) channel partitioning scheme for GSM/GPRS traffic channels; b) introduction of

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1445–1448, 2005.

an auction mechanism for enhancing GPRS services; and c) the proposed model takes into account the value distribution of bidders for revenue maximization.

2 Auction-Based Pricing Model for Admission

In our model, we assume that GPRS users are offered two types of GPRS services: traditional and enhanced services. In traditional GPRS service, GPRS traffic uses the residual channels left from voice traffic while the enhanced GPRS service uses their own dedicated channels. The user who chooses the enhanced service is called a "heavy" user, which refers to his potentially high volume of their traffic in his call requests. To assure that the dedicated channels are allocated to the users who value them most, we employ auction-based prices to admit those users to the dedicated channel (in contrast to PMP, which employs higher prices). Auction rounds are established periodically. Mobile terminals monitor auction information, such as the beginning and the end of an auction, from a base station through the GPRS attached procedure. Users, who are willing to participate in an auction, submit their bids with their call requests through a Packet Data Protocol (PDP) Context Activation. They are required to bid greater or equal to a *reservation price*, which is the minimum price assigned by the network [4]. The winners will be granted transmission by receiving IP address and their data traffic is allowed to use dedicated channels. Since the auction technique is simply used for admitting users to the dedicated channels, the data services are still based on best-effort. However, users can expect better services because of the fact that the data calls would be able to capture relatively more timeslots in dedicated channels than they do in the existing GPRS networks, where resource sharing of voice and data exists. Note that we can reduce user intervention in the auction process by employing agent programs to monitor auctions and automatically submit bids according to user bidding profile, which basically the pattern of bids of users in the current and the next auction rounds (in case it is a losing bid in the first round).

2.1 Assigning Dedicated GPRS Channels

Due to prioritization of voice calls over data calls, the voice blocking probability in GSM/GPRS networks could be approximately determined by the erlang-B formula. However, the number of traffic channels available for voice traffic is now subtracted by the number of dedicated channels. The smaller number of channels for voice traffic would increase the voice blocking probability. Therefore, we need to justify the number of dedicated data channels that should be taken from total traffic channels in order to minimize excessive voice blocking probability. Apparently, the suitable number of dedicated channels depends heavily on the volume of voice calls in the network; we suggest this number should change based on the voice traffic load during the day.

2.2 Maintaining High Throughput

In this subsection, we focus on the performance of the dedicated traffic channel for enhanced data service. It is obvious that each new heavy users in the dedicated channels would impose traffic load that affect the transmission rate of ongoing heavy users. The number of heavy users admitted to the system needs to be limited in order to ensure the welfare of ongoing data transmission in the dedicated channels. We first measure the performance degradation that indicates the excessive number of users. The performance degradation can be measured in terms of *packet delay*. We need to determine a mechanism that allows us to discourage heavy users from entering the system when the average delay is excessive. In auction theory, the parameter used to constrain the amount of bidders participating in an auction is the *auction reservation price* [2]. We employ the reservation price to discourage users from participating in the auction when there are too many users already in the system. The change in reservation price should correspond to capacity of dedicated channels at that particular time, as well as the additional packet delay imposed on ongoing heavy users. The reservation price in an auction starts with a default number, then, price can be increased or decreased in later rounds, based on a target packet delay determined by the operator of the system.

2.3 Optimal Auctions

Our auction-based pricing model is implemented at the admission control level for heavy users. The auction is a multi-unit second-price sealed bid auction or Vickery auction [2]. To maximize revenue gained from the auction, we employ the revenue-maximizing auction or so-called *optimal auction* [4]. The optimal auction in this case is basically a Vickery auction with an introduction of a reservation price. Let us assume that the bidders' private value is independently and identically distributed (IID) drawn from the common distribution $F(v)$: where $F(\underline{v}) = 0$, $F(\overline{v}) = 1$, and $F(v)$ strictly increasing and differentiable over the interval $(\underline{v}\ ,\overline{v})$. We assume further, all bidders are risk neutral. ¿From the revenue equivalent theorem derived by Riley and Samuelson [4], the common equilibrium bidding strategy for Vickery auction yields an expected revenue to the seller as follows:

$$E = n \int_{v_*}^{\overline{v}} (vF'(v) + F(v) - 1)F(v)_{n-1} dv \tag{1}$$

where v_* is the reservation price, n is the number of bidders and $\overline{v}$ is the maximum bidders' value. Even though the revenue equivalent theorem is used to analyze the auction of a single item, the theorem can be generalized for our auction. The auction in our case is a multi-unit auction in which each winner would require only one unit, i.e. the right to access dedicated channels. Hence, we can use the generalization of Vickery auctions, i.e. the k highest bidders would pay the value of the highest losing bid. The revenue-equivalence theorem still holds for these auctions [2]. The reservation price needs to correspond to the private valuation of the network resource. The valuation consists of a *channel cost* and a *congestion cost*. A channel cost refers to the cost of acquiring dedicated channels from existing GPRS channels, i.e., the opportunity cost for partitioning channel. A congestion cost refers to the cost of additional packet delay for admitting heavy users to the dedicated channels. Based on [4], by making the IID assumption and consider risk-neutral bidders, the reservation price, v_* , that maximizes the expected revenue satisfies

$$v_* = v_0 + 1 - \frac{F(v_*)}{F'(v_*)} \tag{2}$$

$$v_0 = A + C \cdot T \tag{3}$$

where v^* is the optimal reservation price, v_0 is the private valuation of the network. A is the channel cost as described earlier; C is a delay-cost factor representing the relationship between the delay in dedicated channels and cost imposed into the system; and T is the average packet delay of the system. The underlined assumption is that additional users would yield additional traffic load in the system, which results in an increase of system delay. These cost need to be added into the private valuation of the network in order to reflect the real cost of the data service. Equations 1 and 2 provide very important parameters for our auction design. Note that the reservation price calculated from equation 2 is independent of the number of bidders [2]. Therefore, we can maximize revenue created from the auction model with the minimum number of bidders.

3 Conclusion

In this paper, we have proposed an auction mechanism, which can provide superior data services in GPRS networks. The approach to our proposed mechanism consists of channel partitioning technique and auction mechanism as an admission control of our system.

References

1. Odlyzko, A. M.,: Paris Metro Pricing: The minimalist differentiated services solution. In: Proc. 1999 7th International Workshop on Quality of Service (IWQoS'99) (1999) 159-161
2. Courcoubetis, C., Weber, R.,: Pricing Communication Networks: Economic, Technology and Modeling. Wiley (2003)
3. Araujo, H., Costa, J., Correia, M.L.,: Analysis of a Traffic Model for GSM/GPRS. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, vol.1 (2001) C-124-C-128
4. Riley, J., Samuelson, W.,: Optimal auctions. The American Economic Review, vol.71, issue 3, June (1981) 381-392

Achieving Stability and Fairness in Mobile Ad Hoc Networks[1]

Ahmed M. Mahdy, Jitender S. Deogun, and Jun Wang

Computer Science and Engineering Department,
University of Nebraska-Lincoln, Lincoln,
NE 68588 USA
{amahdy, deogun, wang}@cse.unl.edu

Abstract. In this paper, we present an approach to stable and fair mobile ad hoc networks. Our approach targets desired quality features using customizable clustering algorithms. Simulation results show that significant improvement on network stability and fairness can be achieved using our approach.

1 Introduction

The infrastructure organization of mobile ad hoc networks has received great attention in the last several years. The widely used cluster-based scheme organizes the network into a layered hierarchy. In this organization, the higher the layer that a node belongs to, the more responsibilities a node carries. This architecture was first used in hierarchal routing in [1]. The two layer (i.e. clusterhead based) scheme is the most popular among the layered hierarchal schemes due to its relative simplicity and less overhead compared to schemes with more than two layers. In clusterhead based networks, the nodes are grouped into clusters supervised by clusterheads. Two major issues are defined in this area; cluster formation and clusterhead selection. Cluster formation refers to how the network is divided into clusters. Static and dynamic clustering are the two cluster formation methods discussed in the literature. Static clustering predetermines the clusters shape and size similar to cellular networks. In contrary, dynamic clustering has no fixed organization and depends on the behavior of the nodes. On the other hand, the research on the selection of clusterheads focuses on the development of selection criteria that distinguish the nodes according to some quality measures such as connectivity degree and mobility.

The effect of the clustering techniques on the network performance is commonly evaluated in terms of network stability and fairness (i.e. load balance). Network stability is adversely proportional to the number of clusterhead replacements; the less the number of clusterhead changes is, the more stable the network is. On the other hand, an ideally fair clustering technique uniformly distributes the managerial load over network nodes. The more the number of nodes involved in the management of the

[1] This work was supported, in part, by an NSF EPSCoR grant EPS-0346476 and by a Nebraska Research Initiative grant on high performance wireless networks.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1449 – 1452, 2005.

network is, the fairer the technique is. There is a tradeoff between network stability and fairness. One of the two merits is sacrificed on the account of the other in many cases for the sake of simpler designs. This can significantly deteriorate network performance. Therefore, the clustering technique should be able to strike a tradeoff between stability and fairness in order to achieve better overall performance. We believe that clustering techniques should be adaptable and configurable to seek specific network merits. The same technique should be configurable for maximum stability, maximum fairness, or optimized overall performance

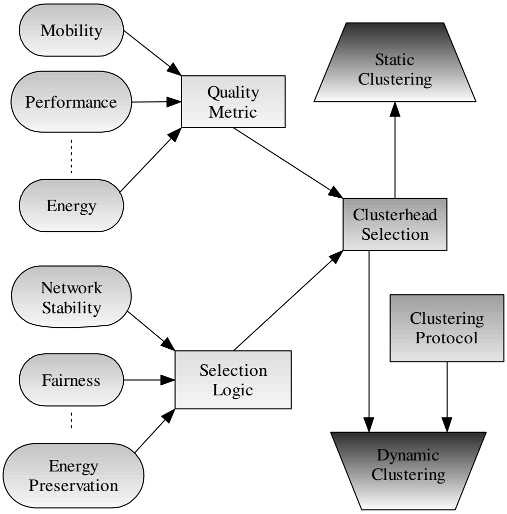


Fig. 1. A block diagram of the clustering framework

2 Stable and Fair Clustering

The management of clusterhead schemes is carried out through a two-step process; cluster formation and clusterhead selection. A clustering protocol regulates cluster formation and coordinates the communication between various nodes. Once the clusters are formed, clusterheads are selected according to some selection criterion.

We propose a clustering framework that generates customizable clustering methods with multi-quality measure selection criteria. The selection logic subcomponent of our framework controls the selection behavior through which the desirable network merits can be achieved. Technically, it is responsible for determining when a new clusterhead should be selected. The number of selection rounds and the frequency of clusterhead replacements greatly affect network stability and fairness. They depend on the logic of the selection algorithm. Accordingly, this subcomponent comprises different selection algorithms that are associated with specific network merits. The desired network merits, therefore, determine which selection algorithm should be used.

How the Framework Works?

1. Identify a set of quality measures.
2. Set relative weights for the chosen measures.
3. Formulate the quality metric, QM.
4. Identify desired network merits.
5. According to the desired merits, choose a selection algorithm, A.
6. Use QM and A to build the Clusterhead Selection component.
7. Use the Clusterhead Selection + Clustering Protocol to form a Clustering Method.

Fig. 2. How the framework works

Ideally, the outcome of the selection process should improve the performance of the cluster as well as the network. Therefore, we evaluate the performance of a selection algorithm based on our framework against the WCA algorithm [2]. Our algorithm is composed of two components, namely Control Loop (CL) and Event Triggered (ET). We designed our algorithm so that it can cover all three possible combinations of stability and fairness by customizing the behavior of the CL and ET components. The CL component focuses on network fairness more than network stability; in contrast the ET component is responsible of achieving high network stability. Our algorithm is designed so that it compromises the tradeoff between the two merits by seeking a midpoint performance.

3 Performance Evaluation

We evaluate the network performance using the following metrics. Number of Switches (NS) is the number of times a clusterhead is replaced. It is an indication of the amount of overhead the system has to afford and a sign of how stable the system is. Maximum Service time Percentage (MSP) is the maximum time a node spends as a clusterhead as a percentage of the total time; a sign of load balance. Energy Standard Deviation (ESD) is the standard deviation of the energy level at the end of the simulation. The results are normalized to the best case scenario where all nodes consume the same amount of energy. This metric reflects the effect of the selection process on the energy consumption and is a sign of system fairness. The larger this measure is, the less fair the system is. We simulate the algorithms performance on a mobile ad hoc network of size 100 to 500 nodes. The results show that our algorithm outperforms WCA achieving improvement of up to 10%, 58%, and 28% on the NS, MSP, and ESD, respectively.

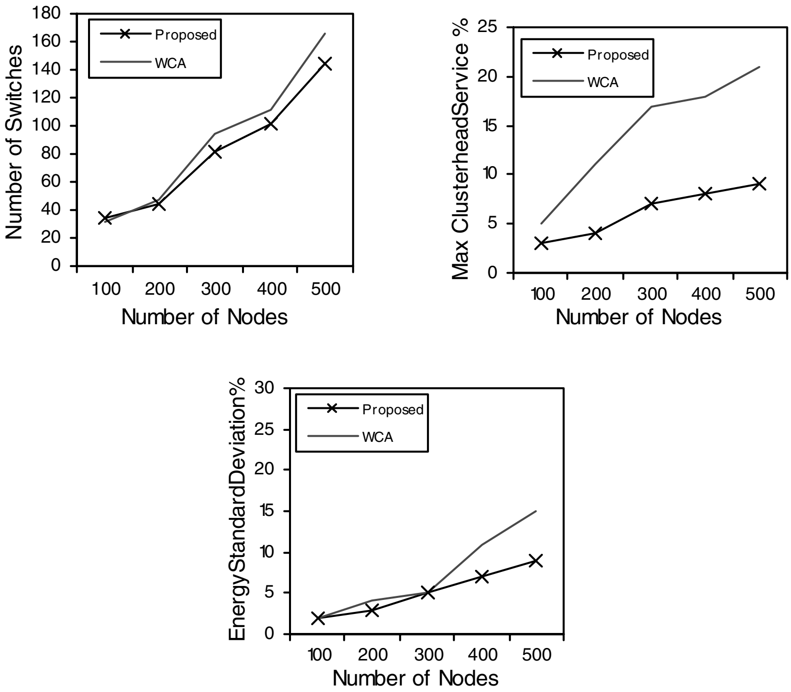


Fig. 2. Performance of the proposed algorithm against the WCA algorithm

References

1. Ephremides, A., Wieselthier, J.E., and Baker, D.J.: A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling. Proc. of the IEEE, 1(1987) 56-73
2. Chatterjee, M., Das, S.K., and Turgut, D.: WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. Journal of Clustering Computing, 2(2002) 193-204

A Novel Method for SCTP Load Sharing

Andreas Jungmaier and Erwin P. Rathgeb

Computer Networking Technology Group,
IEM, University of Duisburg-Essen,
Ellernstr. 29, 45326 Essen, Germany
{ajung, rathgeb}@exp-math.uni-essen.de

Abstract. SCTP is a general purpose transport protocol featuring multi-homing support and flexible, message oriented data delivery services. With respect to multi-homing, the SCTP standard uses this feature for network level redundancy only and, thus, does not provide load sharing capabilities among the redundant links. In order to satisfy the strict performance requirements for signaling transport, efficient load sharing among all active links is desirable in certain scenarios. Therefore, some extensions providing load sharing functionality have been proposed. In this paper, we suggest an improved load sharing algorithm for SCTP with path based selective acknowledgements and present results of a simulation study to demonstrate the benefits of our algorithm.

Keywords: SCTP, transport protocol, multi-homing, load sharing.

1 An Overview of the Stream Control Transmission Protocol

Designed by the Signaling Transport Working Group of the IETF in particular for the transport of signaling data, the Stream Control Transmission Protocol (SCTP) [1] is a general purpose, message-oriented, reliable transport protocol providing a more flexible data delivery service than TCP by using a specific SCTP stream layer, and an increased fault tolerance by allowing network level redundancy through multi-homing. Multi-homed endpoints can be reached by a set of transport addresses rather than only one transport address, therefore multiple distinct paths may exist from an endpoint to its peer endpoint.

SCTP packets consist of a common header, followed by a variable number of information units, which are named *chunks*. There are two types of chunks: control and data chunks. Data chunks contain the actual user messages, while several types of control chunks are used to support the peer-to-peer protocol. The reliable transmission of user data involves data chunks with a 32 bit sequence number (TSN), selective acknowledgement (SACK) control chunks, timers, and multiple selective repeat mechanisms. The amount of data that may be outstanding (i.e. sent but not yet received) is limited by a receiver window regularly announced in the SACK. Flow control parameters (congestion window, *cwnd*, and the slow start threshold, *ssthresh*) are computed for each network path to the

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1453–1456, 2005.

peer. Similarly to TCP, SCTP uses an AIMD algorithm for each path to avoid congestion [1].

The main traffic load is carried by one path only which is the *primary path.* Other paths are only used for data retransmissions and heartbeat control chunks (to monitor the availability of a path). The application may, however, explicitly request to use a path other than the *primary* path for data transmission.

2 A Novel Method for SCTP Load Sharing

SCTP load sharing may be thought of as a periodic change of the primary path. When such changeovers are periodically triggered, significant reordering can be observed by the receiver [2], which is reported to the sender in SACKs containing gap reports [1]. Standard SCTP reacts with unnecessary fast retransmissions, reducing the *cwnd* which unnecessarily limits the overall throughput. Furthermore, standard SCTP only increases the *cwnd* for a path when an incoming SACK from this path advances the highest cumulative TSN acknowledged so far (CTSNA). Therefore, the growth of the *cwnd* of standard SCTP is not adapted to load sharing, happens too slowly and typically mostly for the slowest path. When chunks are delivered to the receiver out of sequence over multiple paths, standard SCTP will send back one SACK for every new incoming data chunk even if no packet loss occurs. Therefore, the rate of returned SACK chunks should be reduced when load sharing is applied. In [2], Iyengar et al. propose the concurrent multipath transfer (CMT) which aims at resolving these problems. This is achieved by (i) avoiding unnecessary fast retransmissions by taking into account the paths to which data chunks were sent and the history of acknowledged chunks when making a retransmission decision; (ii) delaying selective acknowledgements appropriately; (iii) allowing fairer updates of the congestion window, since the *cwnd* for a path should not only be increased, when the CTSNA value for an association is increased by an incoming SACK. Therefore, a *path CTSNA* variable is introduced which stores the highest TSN that was acknowledged for this path without discontinuity. Now the *cwnd* is advanced for paths on which new data chunks were acknowledged and for which the *path CTSNA* has advanced.

Although the CMT algorithm adapts the behaviour of SCTP fairly well to the specifics of a load sharing scenario there is still room for improvement. We therefore propose an algorithm for sending *path based selective acknowledgements* to improve the load sharing performance without significantly increasing the complexity of the algorithms. The basic idea of our proposal is that in addition to the CMT algorithm, the load sharing receiver maintains a SACK counter *path_sack_count* for each path – and not only one per association as with CMT or standard SCTP – and increases this counter by one whenever a packet with data chunks is received on the corresponding path. Whenever *path_sack_count*=2, a SACK is immediately sent over the corresponding path and the counter is reset zero. As a result, two successive SCTP packets with data chunks arriving on different paths can trigger two SACKs on these paths. It should be emphasized, that nevertheless the proposed algorithm still sends one SACK chunk for every two data packets on average in accordance to the requirement in [1]. The strict allocation of SACK chunks to their paths is useful for the SACK-clocking, where each incoming SACK can trigger an update of the outstanding bytes counter, and the receiver and congestion windows.

3 Evaluation of the Novel Algorithm

To evaluate the proposed algorithms, an OPNET [3] simulation model of SCTP was developed which implements the CMT algorithm and our new *path based selective acknowledgements* (PB-SACK) algorithm. Figure 1 shows the simulation scenario with two dual-homed IP-based endpoints connected to routers via gigabit ethernet LAN technology. The routers are interconnected by two bottleneck E3 broadband links with a bandwidth of $W = 34,368$ MBit/s. The delay of Link 1, d_1, was configured to be 10 ms, and the delay of Link 2, d_2, was varied between 10 ms and 200 ms. We assumed a unidirectional data transmission initiated by a saturated traffic source that sends data chunks of constant length (without loss of generality, we assumed 1000 bytes payload per data chunk) towards the receiver.

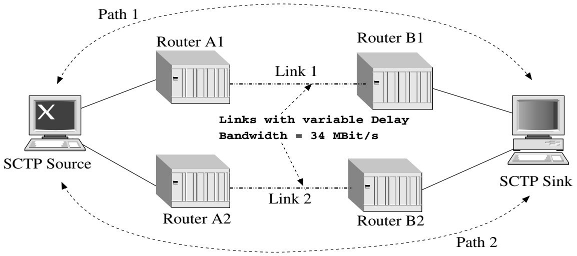


Fig. 1. Simulation scenario

The results presented in Figure 2 are the averaged throughput and maximum message delays as perceived by the receiver application. To isolate the effects due to the load sharing algorithm from those induced by competing traffic in the network, a scenario without interfering background traffic has been used. Due to the fairly deterministic scenario, the confidence intervals calculated from the repeated simulation runs are insignificantly small and have been omitted. The throughput for both algorithms is limited by the bandwidth of the bottleneck links and fully exploits the link capacity for values of $d_2 \leq 20$ ms. Due to the interdependence between transmissions on both links in the loadsharing scenario, the throughput for higher delays of link 2 decreases, even though d_1 remains constant at 10 ms. For 20 ms $< d_2 <$ 70 ms, the throughput of the PB-SACK algorithm is substantially higher than that of the CMT algorithm, as it achieves higher *cwnd* values for path 2. The bandwidth delay product limits the achievable throughput as for all window based transport protocols. Thus, for $d_2 >$ 100 ms,

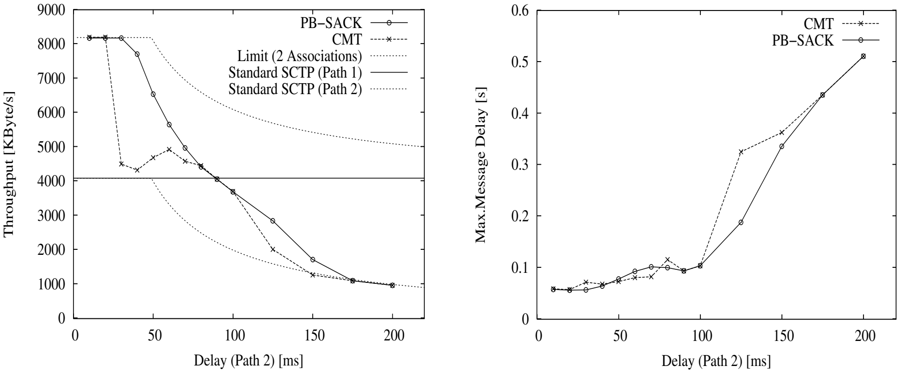


Fig. 2. Application layer throughput and maximum message delays of two load sharing algorithms

the throughput is limited by the receiver window and the link delay. Over the whole parameter range, our PB-SACK algorithm yields a throughput that is higher than or at least as high as that of the CMT algorithm. From the values of the maximum message delays, it is obvious that the increase in throughput of the PB-SACK algorithm has not been achieved at the cost of a higher message delay. For $d_2 > 100$ ms, both algorithms reach a state where effective traffic load distribution cannot be guaranteed any more. At that point, both algorithms allocate traffic almost exclusively to Path 1, and their throughput is limited by a blocked receiver window resulting in an increased message delay.

4 Conclusion

The simulations performed so far have confirmed the benefits of load sharing and the superior performance of our PB-SACK algorithm with respect to the achievable throughput for given delay bandwidth combinations. However, additional simulations in more dynamic scenarios are needed. Moreover, a study on how the extension of the scenario to more than two network paths influences the results could provide some additional insight into to possibilities of SCTP-based load sharing.

References

1. R. Stewart, Q. Xie et al.: RFC 2960 - Stream Control Transmission Protocol, IETF, Network Working Group, October 2000.
2. J.R. Iyengar et al.: Concurrent Multipath Transfer Using SCTP Multihoming, SPECTS 2004, San Jose, July 2004.
3. OPNET Technologies: OPNET Modeler. Commercial simulation tool. Information at http://www.opnet.com/products/modeler/home.html, November 2004

Attaining VoIP–Grade QoS via Deflection: A Buffer Space Tradeoff Study

André Muezerie[1,⋆], Ioanis Nikolaidis[2], and Pawel Gburzyński[2]

[1] Physics Institute of São Carlos - University of São Paulo,
Caixa Postal 369, São Carlos - SP, 13560-970 Brazil
andremuz@if.sc.usp.br

[2] Department of Computing Science - University of Alberta,
Edmonton, Alberta, Canada T6G 2H1
{yannis, pawel}@cs.ualberta.ca

Abstract. We consider the issue of QoS (Quality of Service) in a network catering to isochronous VoIP (Voice over IP) sessions. Our simulation experiments strongly suggest that the single–path paradigm of implementing network–layer virtual circuits neither yields the best utilization of network resources (buffer space) nor is it able to provide the best end-to-end service in terms of loss rate and delay. Instead, it turns out that the dynamic exploration of alternative paths at the packet level with small buffer spaces at the routers results in a more efficient and economical delivery.

Keywords: routing, multiple path routing, deflection, VoIP, QoS.

1 Introduction

Voice-over-IP (VoIP) is quickly becoming a popular application that demonstrates how packet–switched networks can compete in providing services historically associated with circuit–switched reservation–based networks. VoIP sessions pose a wide range of Quality of Service (QoS) requirements, such as low delay, jitter and packet loss, but the anticipated scale of VoIP suggests that fine–grained admission control of the data stream, i.e., at the level of individual calls, would result in a substantial overhead and should be avoided.

A design principle that is "common wisdom" of many QoS solutions is that the most QoS friendly implementation of an end-to-end session should involve a network-layer virtual circuit, whereby all packets of the session follow the same path from source to destination. Intuitively, the deterministic character of such a connection makes it easier to set aside the right amount of resources at every intermediate node and predict what is going to happen when several virtual circuits cross at the same router. Consequently, most work on QoS-driven resource allocation has focused on path selection algorithms [1, 2], assuming that,

⋆ André Muezerie is grateful to CAPES for the generous support of his study leave.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1457–1460, 2005.

once selected, the (single) path will be followed by all packets of the session. This approach equates a transport-layer session with a network-layer virtual circuit, even if (as in the Internet) the network-layer virtual circuit is not explicit. Thus, it does not exploit degrees of freedom afforded by datagram routing.

According to recent results [3], the usefulness of buffering in the core routers for the sake of congestion–relief is highly debatable. In particular, [3] demonstrates that even TCP can live comfortably with a 99% reduction of the buffer space at the routers. These observations indirectly suggest that the intuitions behind the contemporary prevalent trends in QoS provisioning via core router buffer dimensioning are not necessarily correct.

We demonstrate that deflection routing [4] is a viable alternative to providing QoS, due to deflection's flexible ("as needed") use of alternate routing paths compared to routing across pre-established source-destination paths. The results also indicate that real–time isochronous traffic sessions, such as VoIP, *do not* benefit from buffering in the core.

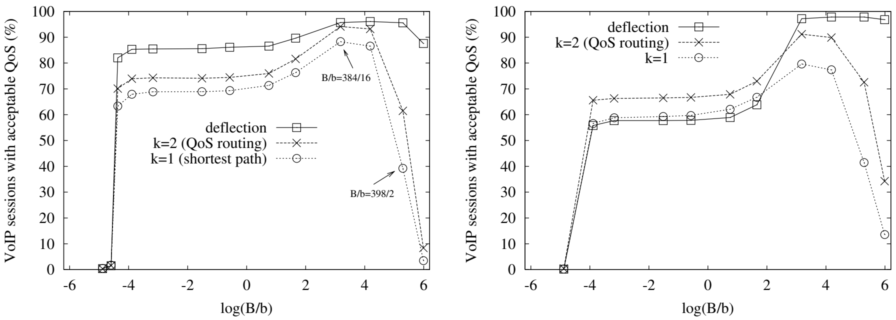
2 The Model

Three basic routing models are considered. In the traditional single path model, packets are forwarded along a single shortest path connecting the source to the destination, or dropped if the output buffer is full. As a variant of traditional routing (and a representative of alternative path routing schemes), we consider a simple strategy whereby the source is aware of a second-best route (disjoint from the first at least on the first hop) and utilizes it whenever the first route appears congested. With deflection routing, each router along the source-destination path is allowed to forward any packet over any of its output links, which are ordered according to the length of the shortest path to the destination offered by the next hop node. The packet is queued for forwarding on the best (i.e., shortest path) link whose buffer is not full.

Each node is capable of acting as a router and a host at the same time, i.e., it can be a source and/or destination of a VoIP session. To express the buffer space tradeoff, we represent the total amount of buffer space in the network as $B + b$, where B denotes the space to be used for reassembly buffers and b stands for the amount of storage equally partitioned among the output links of all routers. The varying ratio B/b determines the adjustable balance between the two categories of storage. While each node implements both types of functionalities, its router and host operations are isolated and they use two separate buffer pools.

3 Results

In the simulated networks, all links have the same bandwidth of 1 Mbps and an equivalent *bandwidth* × *delay* capacity for 2 packets (around 640 km). We set the *playout threshold*, i.e., the delay elapsing between the reception of the first packet of a talkspurt and the actual commencement of the playout to the



(a) 6x5 rectangular topology, 350 simultaneous sessions

(b) triangular topology, 100 simultaneous sessions

Fig. 1. Fraction of sessions with acceptable QoS. VoIP sessions: random source/dest. pairs, bursty talkspurts (mean 1.2 s) and silences (mean 1.8 s), packetization every 20ms, 200 bytes (160 voice + 40 headers)

equivalent of four packet transmissions. We define "acceptable QoS" as a loss rate below 5%, which is tolerated by most audio decoders with no appreciable degradation in voice quality.

Interestingly, it turns out that extending the reassembly buffer at the destinations in proportion to the reduction of the buffer space at the routers is an over-compensation. Consider Fig. 1(a), showing the percentage of sessions with acceptable QoS regardless of the actual size of the reassembly buffer at the destination for a rectangular topology (similar to the bi-directional Manhattan Street Network [5], except for the absence of the links closing the rectangle onto a torus). The curves $k = 1$ correspond to the classic single path routing and $k = 2$ represents the case where the source has two distinct shortest paths to the destination.

With deflection, we observe a trend of improving QoS as buffer space is removed from the routers and assigned to the destinations. In traditional circuit-style routing, in the absence of deflection, the routers require storing the packets that cannot be immediately forwarded on the single congested output link. Figure 1(a) in fact demonstrates that the benefits (flexibility) of deflection more than compensate for the reduced buffer space at the routers. At the area of large B/b, where the traditional routing breaks down, deflection is still able to provide reasonable service to a large population of sessions. In agreement with the observations made in [5], deflection can take advantage of some limited router buffer space. In comparison to a completely buffer-less case, small buffers bring considerable improvement, while large buffers yield no appreciable gains.

Even small networks can benefit from deflection. In the particular topology shown in Fig. 2, the best (shortest) routes for all nodes pass through the even nodes ("hotspots"), and when the links between these nodes become congested, they aggressively discard packets under classical routing (Fig. 1(b)). With de-

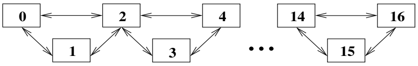


Fig. 2. Triangular topology

flection the links between even and odd nodes are used to distribute the load and achieve higher throughput and therefore, lower losses. At higher loads the effect is even more pronounced.

4 Conclusions

We analyzed the QoS perceived by concurrent VoIP sessions in networks operating under three routing policies: 1) classic shortest path best effort routing, 2) QoS routing with two alternative shortest paths, and 3) asynchronous deflection. Our experiments show that, especially under high loads, deflection performs at least as well as the other two alternatives, and frequently outperforms them with respect to the overall QoS measures perceived by the application.

The better load distribution over the network brought about by deflection translates into better buffer space utilization as a global network resource, lower losses and end-to-end delays. Contrary to common belief, deflection is not necessarily inferior (from the viewpoint of service guarantees) to the more deterministic alternatives involving pre-arranged predictable routes. Despite the difference in appearance, neither paradigm is in fact deterministic and predictable, and the application-level perception of its operation does not necessarily endorse determinism within the core.

References

1. Apostolopoulos, G., Guerin, R., Kamat, S., Tripathi, S.K.: Quality of service based routing: a performance perspective. In: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, ACM Press (1998) 17–28
2. Xiao, W., Soong, B.H., Law, C.L., Guan, Y.L.: Evaluation of heuristic path selection algorithms for multi-constrained qos routing. In: IEEE International Conference on Networking, Sensing and Control. Volume 1. (2004) 112–116
3. Appenzeller, G., Keslassy, I., McKeown, N.: Sizing router buffers. In: ACM SIGCOMM'04, Portland, Oregon (2004)
4. Olesinski, W., Gburzynski, P.: Service guarantees in deflection networks. In: Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01). (2001) 267–274
5. Borgonovo, F., Cadorin, E.: Locally-optimal deflection routing in the bidirectional Manhattan network. In: Proceedings of IEEE INFOCOM'90. (1990) 458–464

Applying the DiffServ Model to a Resilient Packet Ring Network

Fredrik Davik[1,2,3] and Stein Gjessing[1]

[1] Simula Research Laboratory
[2] University of Oslo
[3] Ericsson Research Norway
{bjornfd, steing}@simula.no

Keywords: Resilient Packet Ring, Differentiated Services, Per Hop Behavior.

1 Introduction

In this paper we introduce a formal specification of parts of the service differentiation mechanisms of the recent IEEE 802.17 Resilient Packet Ring (*RPR*) standard [1, 2, 3], and assess RPR's suitability for use in a DiffServ environment. We propose a simple mapping between RPR's traffic classes and three standardized DiffServ Per Hop Behavior groups. When using this mapping, we discuss, by use of an analytical example and simulation results, the behavior of the traffic assigned to each PHB group in terms of its throughput (all PHBs) and delay properties (for the EF PHB only). For the simulation part, we use our OPNET [4] implementation of the RPR standard. In the full length version of this paper, for the proposed mapping, we analyze conformance to the requirements specified for the three DiffServ PHBs [5].

2 Delay Guarantees and Rate Control in RPR

In an RPR node, the *Rate Control Block* imposes several per ringlet and per node rate constraints on local ingress (add)- and transit traffic. In this chapter, we introduce a set of invariants that expresses the effect of the RPR scheduling constraints, before illustrating this using an analytical example. We use the notation R_X to specify the rate constraint in effect for a particular RPR traffic class, where $X \in \{A, A0, A1, B, C\}$. *offered(X)* represents the amount of traffic of a particular traffic class, X, that a node or a set of nodes **want(s)** to transmit. *accepted(X)* refers to the corresponding amount of traffic that **will** be transmitted. R_L is the link rate (all links in an RPR network have the same rate).

For class A ($A0$ and $A1$) traffic, the per-node amount of traffic is constrained, regardless of the destination of the traffic transmitted, as shown in invariants 1 and 3-6. For class B and class C traffic, we use per link invariants as shown in invariants 7-10.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1461–1464, 2005.

invariant 1 $R_R = \sum_{j \in \{nodes\}} R_{A0_j} \leq R_L$

invariant 2 $R_U = R_L - R_R$

invariant 3 $\sum_{j \in \{nodes\}} \left(R_{A1_j} + R_{B_j} \right) \leq R_U$

invariant 4 $\forall nodes : \mathit{offered(A)} \leq R_{A0} \Rightarrow \mathit{accepted(A0)} = \mathit{offered(A)}$

invariant 5 $\forall nodes : R_{A0} < \mathit{offered(A)} \leq R_{A0} + R_{A1} \Rightarrow$
$\mathit{accepted(A0)} = R_{A0} \wedge \mathit{accepted(A1)} = \mathit{offered(A)} - R_{A0}$

invariant 6 $\forall nodes : \mathit{offered(A)} > R_{A0} + R_{A1} \Rightarrow$
$\mathit{accepted(A0)} = R_{A0} \wedge \mathit{accepted(A1)} = R_{A1}$

invariant 7 $\forall links : \mathit{offered(B)} + \mathit{accepted(A1)} \leq R_U \Rightarrow \mathit{accepted(B)} = \mathit{offered(B)}$

invariant 8 $\forall links : \mathit{offered(B)} + \mathit{accepted(A1)} > R_U \Rightarrow$
$\mathit{accepted(B)} = R_U - \mathit{accepted(A1)}$

invariant 9 $\forall links : \mathit{offered(C)} + \mathit{accepted(A1)} + \mathit{accepted(B)} \leq R_U \Rightarrow$
$\mathit{accepted(C)} = \mathit{offered(C)}$

invariant 10 $\forall links : \mathit{offered(C)} + \mathit{accepted(A1)} + \mathit{accepted(B)} > R_U \Rightarrow$
$\mathit{accepted(C)} = R_U - \mathit{accepted(A1)} - \mathit{accepted(B)}$

The DiffServ Expedited Forwarding (EF) PHB is a DiffServ building block to be used for the provisioning of services that provides low loss, low delay and low jitter. Another DiffServ PHB, namely the Assured Forwarding (AF) PHB is a specification of a PHB group, that may contain up to four independent AF classes. Each class must be allocated a separate amount of forwarding resources. Within each AF class, an implementation must provide a minimum of two and a maximum of four different drop probabilities. The DiffServ default PHB is specified as a best-effort (BE) forwarding behavior. For the reminder of the paper, we refer to the default PHB as the BE PHB. Given the detailed requirements associated to the EF, AF and BE PHBs discussed in detail in [5], and the properties of RPR's

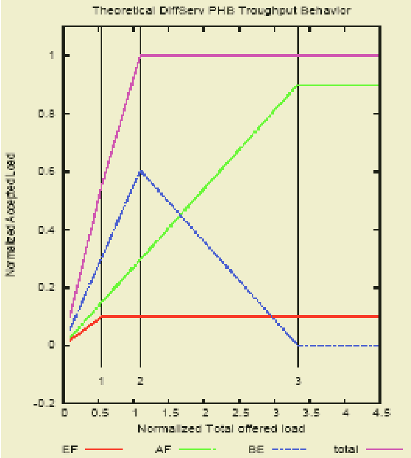
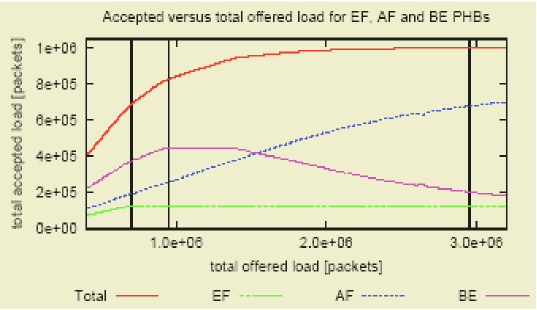


Fig. 1: Theoretical Throughput for the EF, AF and BE PHBs on a single link, in an RPR network

traffic classes A, B and C as defined by the invariants; we propose that the DiffServ EF, AF and BE PHBs are mapped to respectively RPR traffic classes A, B and C. In figure 1, we have shown the analytical resulting throughput when using this mapping and enforcing the invariants introduced above. The figure shows the analytical result, using the offered load as the free variable with a 55/27/18 ratio for respectively BE, AF and EF offered traffic, and plotting the corresponding values for accepted EF, AF and BE traffic as well as the aggregate of these values. Three points for offered load in this plot are of particular interest. As long as we are on the left side of point 1, the aggregate EF traffic can be increased further while maintaining invariant 4 or 5. At point 1, we have reached the load level where an additional load increase causes the premise of both invariants 4 and 5 to be false. Beyond this point, the maintenance of invariant 6 effectively prohibits the acceptance of additional EF traffic. On the left side of point 2, as long as invariant 9 is maintained, an increase in offered BE traffic leads to an equivalent increase in accepted BE traffic. At point 2, we have reached the load level where an additional load increase causes the invariant's premise to be false. Beyond this point, the maintenance of invariant 10 effectively prohibits the acceptance of more BE traffic. Finally, between points 2 and 3, as long as invariants 7 and 10 are maintained, an increase in offered AF traffic leads to an equivalent increase in accepted AF traffic, on the expense of an equivalent decrease in accepted BE traffic. At point 3, we have reached the load level where an additional load increase causes the premise of invariant 7 to be false. Beyond this point, the maintenance of invariants 8 and 10 effectively prohibits the acceptance of additional AF traffic as well as excluding BE traffic from the link.

3 Performance Evaluation by Simulations

We present results obtained for a 16 node ring, where the link delay and capacity is respectively 250 μs and 1Gbit/s for all links (i.e. a 800km metro ring). For



(a)

hops	median	$\overline{x}$	σ	max	n
1	0.0	0.6	1.3	24.1	12633535
2	0.0	1.2	1.7	24.4	13060494
3	1.2	1.8	2.0	27.0	13581686
4	2.0	2.4	2.3	24.0	14681577
5	2.3	2.7	2.6	32.9	15525550
6	2.9	3.3	2.8	28.6	15241110
7	3.9	4.3	3.1	28.0	10212709
8	4.9	5.3	3.4	30.0	4283579

(b)

Fig. 2: a) Accepted vs. total offered load for EF, AF and BE PHBs. b) EF traffic delay statistics (in units of μs). Per link propagation and transmission delays are excluded. The min value is 0 for all hop counts

each simulation, an offered load parameter is given. 101 load values are used, linearly distributed on a scale from 1 to 10 times the base load. For each load value, 16 simulations are executed with different sets of source destination pairs. The base load uses the same ratio between the PHBs as specified in section 2. All nodes transmit traffic of each PHB group using a Poisson distribution. Figure 2a plots the offered vs. accepted load for the EF, AF and BE PHBs, as well as the aggregate of all PHBs. Each point on the curves is the mean of the throughput values obtained from the 16 different sender-receiver simulations for that particular offered load. The vertical lines in the figure corresponds roughly to the points discussed for the analytical results in the previous section. When looking at the throughput-performance of the ring, the behavior is not as clear-cut as for the theoretical behavior, illustrated in figure 1. In a real (or simulated) RPR network, the individual links become congested at different points in time and at various levels of per-node offered load. Thus for a given sender-receiver pair setup, the effect of the invariants discussed in the previous section, will occur at at different values of per node offered load for the various congested links on the ring. The effect of the invariants is additionally concealed as we take the mean of the values obtained from a number of different sender-receiver pair configurations. However, the general trend from figure 2a clearly resembles that of the analytical example. Figure 2b show the EF-traffic delay statistics broken down on a per-hop level. The mean values are in the range $[0.6, 5.3]\mu s$, which is well within the transmission time of two 500B packets. The standard deviation is very low (less than transmission time of a 500B packet). As expected, the median, mean, standard deviation and max values all increase as a the number of hops increase for the samples observed. But as shown, the jitter is very low and in the worst case within 32.9 μs.

4 Conclusion

In this short paper, we have proposed and evaluated, analytically and by simulations, a mapping between the three DiffServ PHBs: Expedited Forwarding (EF), Assured Forwarding (AF) and BE (Default) and the RPR traffic classes: A, B and C. Our results indicate, that the proposed mapping scheme appear reasonable and that conformance to DiffServ's PHB requirements appears to be possible.

References

1. IEEE Computer Society: IEEE Std 802.17-2004 (2004) 1
2. Davik, F., Yilmaz, M., Gjessing, S., Uzun, N.: IEEE 802.17 Resilient Packet Ring Tutorial. IEEE Commun. Mag. **42** (2004) 112–118 1
3. Gambiroza, V., Yuan, P., Balzano, L., Liu, Y., Sheafor, S., Knightly, E.: Design, analysis, and implementation of DVSR: a fair high-performance protocol for packet rings. IEEE/ACM Trans. Networking **12** (2004) 85–102 1
4. (OPNET Modeler. http://www.opnet.com) 1
5. Davik, F., Gjessing, S.: Applying the DiffServ Model to a Resilient Packet Ring Network. Technical Report 2005-1, Simula Research Laboratory (2005) http://www.simula.no/publication.php. 1, 2

Maximizing the Number of Connections in Multifiber WDM Chain, Ring and Star Networks[*]

Katerina Potika

Computer Science, ECE, National Technical University of Athens, Greece
epotik@cs.ntua.gr

Abstract. We study a wavelength assignment maximization problem in multifiber WDM networks. Given a network G, a set of requests, the number of fibers per link and the number of wavelengths, we want to maximize the number of requests that can be satisfied simultaneously. We propose polynomial time algorithms, that are either optimal or have a guaranteed worst case performance in basic topologies, such as chain, ring and star networks.

1 Introduction

A recent advance in WDM technology is the use of multiple fibers per link. In such networks signals that pass through the same link may use the same wavelength under the constraint that they traverse different parallel fibers.

We are given a network $G = (V, E)$, a function $\mu : E \rightarrow \mathbb{N}$, that defines the multiplicity of fibers on each link, a set of paths $\mathcal{P}$ (each request is prerouted) and w colors. A path can be satisfied if a wavelength (color) is assigned along that path (coloring). A path multicoloring for $(G, \mathcal{P}, \mu, w)$ is valid, if for each edge e, paths from $\mathcal{P}$ that pass through e are colored so that any color, out of the w, is used at most $\mu(e)$ times.

We study problem Maximum Path Multicoloring (Max-PMC). *Given a graph $G = (V, E)$ with edge weights $\mu : E \rightarrow \mathbb{N}$, a set of paths $\mathcal{P}$ and a number w, find a valid path multicoloring of a subset $\mathcal{P}' \subseteq \mathcal{P}$ with w colors such that $|\mathcal{P}'|$ is maximized.*

A well known problem, that is a simplified version of our Max-PMC, is the Maximum Integral Multicommodity Flow (Max-MF). *Given a graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{N}$ and a set of flows $\mathcal{P}$ find a valid, with respect to c, path packing of $\mathcal{P}$, i.e., a subset $\mathcal{P}' \subseteq \mathcal{P}$ such that no more than $c(e)$ paths in $\mathcal{P}'$ go over edge e. The goal is to maximize $|\mathcal{P}'|$.*

Problem Max-PMC in rings and stars is $\mathcal{NP}$-hard because the single fiber problem Max-PC is $\mathcal{NP}$-hard in rings ([1]) and stars ([2]). For $\mathcal{NP}$-hard prob-

[*] Research supported in part by the Greek Ministry of Education through grants "Pythagoras" and "Protagoras", of the EU funded program EPEAEK.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1465–1470, 2005.

lems we use approximation algorithms. An algorithm A for a maximization problem Π is a ρ-approximation (for $\rho > 1$) if for every instance I of Π, A runs in time polynomial in $|I|$ and delivers a solution with cost $SOL \geq 1/\rho \cdot OPT$, where OPT denotes the cost of an optimal solution for I.

Related Work

For Max-PMC in trees a 2.54-approximation algorithm is presented in [3]. Saad and Luo in [4] study problem Max-PMC without assuming a special topology. They propose a well known method that considers only one wavelength at a time, and can be iteratively used for any number of wavelengths. In [5] they solve Max-PMC using two heuristics in arbitrary topologies with uniform number of available fibers per link.

An interesting related problem for single-fiber networks is the Maximum Path Coloring Problem (Max-PC). In chains it can be solved exactly [6]. Algorithms for Max-PC in rings with 3/2-approximation is given in [1] and in stars with 1.58-approximation [2].

Our results

In chain networks we prove that Max-PMC can be solved optimally. In ring networks we present for Max-PMC a $1 + \frac{1}{\mu_{min}-2}$-approximation algorithm, where μ_{min} is the minimum edge multiplicity that occurs in the ring network. Furthermore, we show that the existence of an exact algorithm for the Max-MF for a network implies a $1/(1 - e^{-1})$-approximation algorithm for Max-PMC in this network. Applying this algorithm for ring and star networks we obtain two 1.58-approximation algorithms.

Technical Preliminaries

A multifiber network can be modeled by an undirected graph $G = (V, E)$. We use $n = |V|$ and $m = |\mathcal{P}|$. For a set $\mathcal{P}$ and an edge e we denote by $L(e, \mathcal{P})$ the *load* of edge e, that is the number of paths in $\mathcal{P}$ that use e. The number of paths that can be colored in each edge is at most $\mu(e) \cdot w$. If we denote by $\mathcal{P}'$ the subset of satisfied paths then the following condition must hold in each edge e:

$$L(e, \mathcal{P}') \leq \mu(e) \cdot w \tag{1}$$

An upper bound of the optimal solution of an instance $I = (G, \mathcal{P}, \mu, w)$ for Max-PMC is the optimal solution of an instance $I' = (G, \mathcal{P}, \mu \cdot w)$ for Max-MF:

$$OPT(I) \leq OPT_{\text{Max-MF}}(I') \tag{2}$$

We will also use as a subroutine an algorithm in chains for problem Path Multicoloring with Minimum Number of Collisions (MinCollisions-PMC). In this problem we are given a set of paths $\mathcal{P}$ and the number of wavelengths, and we want to satisfy all paths in $\mathcal{P}$ by finding for every edge an appropriate fiber multiplicity. Problem MinCollisions-PMC in chains can be solved optimal by using algorithm [7].

2 Chains

For chains we propose Algorithm 1. We can solve problem Max-MF in chains optimally using a algorithm described in [8] in $O(n+m)$ time.

Algorithm 1. Max-PMC in chains
Input: chain $G=(V,E)$, a function $\mu : E \to \mathbb{N}$, a set of paths $\mathcal{P}$ and w colors.
Output: a maximum subset $\mathcal{P}' \subseteq \mathcal{P}$ and a multicoloring of $\mathcal{P}'$

```
1. Call algorithm for Max-MF in chains for instance $(G, \mathcal{P}, \mu \cdot w)$, denote the returned
   solution by $\mathcal{P}'$
2. Color instance $(G, \mathcal{P}', w)$ with algorithm for MinCollisions-PMC in chains
3. Return a multicoloring of the set of paths $\mathcal{P}'$
```

Theorem 1. *Algorithm 1 solves optimally problem* Max-PMC *in chains.*

Proof. We obtain a valid multicoloring of all paths in $\mathcal{P}'$ using no more than w colors. Algorithm for problem MinCollisions-PMC in chains returns an optimal solution, i.e. in each edge e, the number of paths in $\mathcal{P}'$ that pass through e and have the same color, is at most $\lceil \frac{L(e,\mathcal{P}')}{w} \rceil$. Using condition (1) we have that the number of color repetitions in each edge e is $\lceil \frac{L(e,\mathcal{P}')}{w} \rceil \leq \lceil \frac{\mu(e)\cdot w}{w} \rceil \leq \mu(e)$, because $\mu(e)$ is an integer. Thus, the available multiplicities $\mu(e)$ suffice and in each edge we color $L(e,\mathcal{P}')$ paths.

3 Rings

The proposed algorithm is Algorithm 2. Algorithm for Max-MF in rings [8] gives an optimal solution, denoted by $\mathcal{M}$.

Theorem 2. *Algorithm 2 for* Max-PMC *in rings is a* $\frac{\mu_{min}-1}{\mu_{min}-2}$*- approximation algorithm, where* $\mu_{min} = \min_{e \in E} \mu(e)$.

Proof. We first have to show that set $\mathcal{M} - \mathcal{P}_0$ can be valid multicolored. Algorithm transforms the ring in a chain by 'unfolding' the ring in node v_0. Doing that the load on some edges will be split in two parts, but on these edges we removed at most w paths (see Figure 1). Instead of one edge e in the ring instance we may have two edges e' and e'' in the chain instance. In the edges e, in which the load is split, the number of times each color is repeated is at most:

$$\lceil \frac{L(e',\mathcal{M})}{w} \rceil + \lceil \frac{L(e'',\mathcal{M}) - L(e,\mathcal{P}_0)}{w} \rceil \leq \lceil \frac{L(e',\mathcal{M})}{w} \rceil + \lceil \frac{L(e'',\mathcal{M}) - w}{w} \rceil \leq$$
$$\lceil \frac{L(e',\mathcal{M}) + L(e'',\mathcal{M})}{w} \rceil + 1 - 1 \leq \lceil \frac{L(e,\mathcal{M})}{w} \rceil \leq \lceil \frac{\mu(e)w}{w} \rceil \leq \mu(e)$$

The two last lines of the above equation holds because of condition (1) and because $\mu(e)$ is an integer. From equation (2) and from the facts that the solution

Algorithm 2. Max-PMC in rings

Input: ring $G = (V, E)$, a function $\mu : E \rightarrow \mathbb{N}$, a set of paths $\mathcal{P}$ and w colors.

Output: a maximum subset $\mathcal{P}' \subseteq \mathcal{P}$ and a multicoloring of $\mathcal{P}'$

```
1. Call algorithm for Max-MF in rings for instance $(G, \mathcal{P}, \mu \cdot w)$, denote the returned solution, by $\mathcal{M}$.
2. Each edge $(v_i, v_{i+1})$ is denoted by $e_i$. Find node $e_0$ with minimum multiplicity and reindex nodes accordingly.
3. Find the paths in $\mathcal{M}$ that pass through node $v_0$, denote them by $\mathcal{Q}$. Sort the paths in $\mathcal{Q}$ in descending order of the index of their clockwise endpoint and choose the $w$ first paths, denote them by $\mathcal{P}_0$.
4. Set $\mathcal{P}' = \mathcal{M} - \mathcal{P}_0$.
5. Transform the ring $(G, \mathcal{P}')$ to a chain $(G', \mathcal{P}'')$ as follows:
   a. The chain graph $G'$ consists of $n + s + 1$ nodes, namely $v'_0, \ldots, v'_{n+s}$.
   Let $e'_i = (v'_i, v'_{i+1}), 0 \leq i \leq n + s - 1$.
   b. For each path $\langle v_i, v_j \rangle \in \mathcal{P}'$, add a path to $\mathcal{P}''$:
   if $i < j$ add path from node $v'_i$ to node $v'_j$ to $\mathcal{P}''$, otherwise add path form node $v'_i$ to node $v'_{n+j}$ to $\mathcal{P}''$.
6. Call Algorithm for MinCollisions-PMC in chains on instance $(G', \mathcal{P}'', w)$. Color each path in $\mathcal{P}'$ with the color of the corresponding path in $\mathcal{P}''$.
```

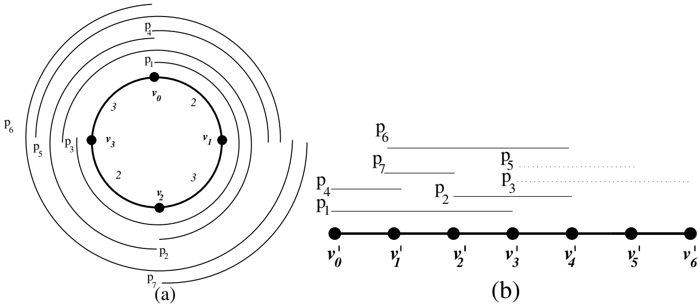


Fig. 1. An example of how we transform the ring (a) into a chain (b). In (a) we see all the paths in $\mathcal{M}$ and in each edge the available multiplicity. Lets assume that $w = 2$. Then $\mathcal{P}_0 = \{p_3, p_5\}$ and we multicolor the remaining paths p_1, p_2, p_4

is $SOL \geq |\mathcal{M}| - |\mathcal{P}_0|$, $\mathcal{P}_0 \leq w$ and $\mathcal{M} \geq (\mu_{min} - 1)w$ (where $\mu_{min} = \min_{e \in E} \mu(e)$), we have that the following holds:

$$\frac{SOL}{OPT} \geq \frac{|\mathcal{M}| - |\mathcal{P}_0|}{|\mathcal{M}|} \geq 1 - \frac{\mathcal{P}_0}{\mathcal{M}} \geq 1 - \frac{w}{(\mu_{min} - 1)w} = \frac{\mu_{min} - 2}{\mu_{min} - 1} \tag{3}$$

Thus we have a $\rho = \frac{\mu_{min} - 1}{\mu_{min} - 2} = 1 + \frac{1}{\mu_{min} - 2}$ Observe that for $\mu_{min} \geq 4$, we obtain an approximation ratio $\rho \leq 3/2$.

4 Iterative Algorithm for Rings and Stars

Our algorithms use again as a subroutine algorithms for the Max-MF problem. Initial we have an empty set $\mathcal{P}'$. For each color i ($0 \leq i \leq w$) we find a valid with respect to μ path packing $\mathcal{P}_i \subseteq \mathcal{P}$. Every path in $\mathcal{P}_i$ takes color i, is removed from $\mathcal{P}$ and is added to $\mathcal{P}'$. The proof of Theorem 3 is omitted.

Theorem 3. *The iterative algorithm for* Max-PMC *is a* $1/(1-e^{-1})$*- approximation if an exact algorithm for* Max-MF *is used as a subroutine.*

When we apply the iterative algorithm for Max-PMC in rings we get another algorithm.

Corollary 1. *The iterative algorithm for* Max-PMC *in rings is a* 1.58*-approximation algorithm.*

Algorithm 2 for Max-PMC in rings has worse approximation for instances of the problem where $\mu_{min} < 4$ than the iterative one, but for instances with $\mu_{min} \geq 4$ the approximation performs better, i.e. $3/2 \geq \rho \geq 1$. If μ_{min} is very large our algorithm is near optimal, with ρ close to 1.

Problem Max-MF can be solved optimally in stars (see [9]), thus we have:

Corollary 2. *The iterative algorithm for* Max-PMC *in stars is a* 1.58*-approximation algorithm.*

5 Conclusions and Open Problems

In this paper we present algorithms for Max-PMC in chains, rings and stars. In rings the solution can be as good as optimal if the minimum fiber multiplicity of the network is large enough, thus improving the proposed algorithm for general networks in [4]. All our algorithms are easy to implement, run in polynomial time and have guaranteed worst case behavior. Some open problems for future work include the variation where we we are given requests (pair of nodes) instead of paths. Another interesting version of Max-PMC is the one, where each request has a profit and the goal is to maximize the total profit of satisfied requests.

References

1. Nomikos, C., Pagourtzis, A., Zachos, S.: Satisfying a maximum number of prerouted requests in all-optical rings. Computer Networks **42** (2003) 55–63
2. Erlebach, T., Jansen, K.: Maximizing the number of connections in optical tree networks. In: Proceedings of the 9th Annual International Symposium on Algorithms and Computation (ISAAC '98). LNCS 1533 (1998) 179–188
3. Erlebach, T., Pagourtzis, A., Potika, K., Stefanakos, S.: Resource allocation problems in multifiber WDM tree networks. In: Proc. of the 29th Workshop on Graph Theoretic Concepts in Computer Science. LNCS 2880, Springer-Verlag (2003) 218–229

4. Saad, M., Luo, Z.: On the routing and wavelength assignement in multifiber WDM networks. In: Proceedings of Globecom 2002. (2002)
5. Li, D., Jia, X., Hu, X., Gong, E.Z.: Wavelength assignment for minimizing system blockings in multifiber all-optical WDM networks. Optical Networks **4** (2003) 75–81
6. Carlislie, M., Lloyd, E.: On the k-coloring of intervals. Discrete Applied Mathematics **59** (1995) 225–235
7. Nomikos, C., Pagourtzis, A., Zachos, S.: Routing and path multicoloring. Information Processing Letters **80** (2001) 249–256
8. Adamy, U., Ambuehl, C., Anand, R.S., Erlebach, T.: Call control in rings. In: Proceedings of the 29th International Colloquium on Automata,Languages, and Programming (ICALP 2002). LNCS 2380 (2002) 788–799
9. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica **18** (1997) 3–20

A Hierarchical Secure Ring-Oriented Multicast Protocol over Mobile Ad Hoc Network*

Choong Seon Hong and Yubai Yang

Department of Computer Science, Kyung Hee University,
1 Seocheon, Giheung, Yongin, Gyeonggi 449-701 Korea
cshong.khu.ac.kr, yyb@networking.khu.ac.kr

Abstract. In this paper, we propose a novel scheme of Hierarchical Eulerian Ring-Oriented Multicast Protocol over mobile ad hoc network. It has features that concentrate on efficiency and robustness simultaneously. It is also an application-driven proposal for hazard detection. Simulation results show different level of improvements on control traffic, end-to-end delay by comparing with tree-based and mesh-based multicast protocols.

Keywords: Hierarchical, Eulerian Ring, Multicast, Ad hoc network.

1 Proposed Scheme

We employ Fisheye State Routing (FSR) protocol as our IP protocol, with which we can get knowledge within predefined-scope. FSR introduces the notion of multi-layer fisheye scope to reduce routing update overhead in large-scale networks. Mobile nodes exchange link state entries with their neighbors with a frequency that depends on distance to destination. With the help of FSR as IP protocol, our proposal can get the necessary information within predefined scope that is the basis of searching algorithm for hierarchical Eulerian ring.

1.1 A Novel Eulerian Ring Based Scheme

Let G be a planar graph, stands for a group of mobile nodes in ad hoc network. Nodes in G are denoted by V(G); links between nodes in G are denoted by E(G) (we discuss only bi-direction link herein); The degree of V is the number of edges meeting at V, and is denoted by deg v. If there is a closed trail that includes every edge once and only, such a trail is called a Eulerian trail. If it forms a ring, it is called a Eulerian ring. Proofs of theorems are omitted, details could be found in [2]. The reason that we employ Eulerian Ring is edge-traceable. In Eulerian graph we could walk along every link once and only once. First we make the following definitions for terminology and rule: Theorem 1: let G be a connected

* This work was supported by University ITRC Project of MIC, Dr. C.S.Hong is the corresponding author.

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1471–1474, 2005.

graph, then G is Eulerian if and only if every vertex of G has even degree. Center node: who provides FSR link-state information of the region. Multicast receiver: node who receives multicast packets. $E_{i,j}$: a link connects node i and node j. Leaf node: a node whose deg v=1. Even node: node with even degree. Odd node: node with odd links. Rule 1: let neighbor take leaf node's task (note that leaf node has only one neighbor), then delete leaf nodes. Execute repeatedly until there is no leaf node in G. Calculate degree then mark node type (even node and odd node). 2-odd-link: link that connects 2 odd nodes. 2-even-link: link that connects 2 even nodes. Odd-even-link: link that connects one odd node and one even node. Rule 2: mark link type (2-odd-link, 2-even-link or odd-even-link). Maintain 2-even-link unchanged, prune 2-odd-link in descending degree (from maximum degree to minimum degree), and is denoted by freeze-link. Execute rule 2 repeatedly until no 2-odd-link exists. Theorem 2: In a graph G, the number of nodes of odd degree is even. After rule 2, only odd-even-links remain. And odd nodes will appear in pairs Rule 3: Mark odd node pairs in descending degree. Find path connected them in descending way, mark it recursive-freeze-link and prune it. If new odd node generates, execute rule 3 recursively until no odd-node-pair exists. Main ring: ring contains center node. Assist ring: ring other than main ring but connects to main ring. It receives multi-cast packets from diverge nodes. Branch: a tree or line connects to ring. It contains at least one multicast receiver and have a joint to a ring. Master: a node that takes charge in multicast service for his ring or branch. Each ring, assist ring and branch should have its own master respectively. The master of assist ring is also referred as sub-master. Member: node that takes part in multicast affairs, including multicast member and forwarding node. Diverge node: a node resides on joint of rings or joint of ring and branch. It duplicates multicast packet for the lower ring or branch. Bridge: the only link without which graph will be disconnected. δ: A time-out setting to prevent situations in which a node has waited too long for a token. For example, due to crash failure of node or link, a message taking time unites longer than for transmission is considered as having been lost. Theorem 3: if G is a graph in which every vertex has even degree, then G can be spitted into disjoint cycles-that is, no two cycles have any edges in common. Rule 4: execute Fleury algorithm [2], then split it into small circles according to theorem 3; construct ring-based topology by marking main ring, assist ring and branch. Recover the freeze-link and recursive-freeze-link, make them part of branch or assist ring if it contains member, and so does the leaf node generated in Rule 1.

1.2 Hierarchy

The second character in the protocol is hierarchy, which is achieved by topology of different level Eulerian rings and branches. The hierarchical architecture has 2 advantages, multicast routing updates and control traffic could be restricted locally to reduce delivery overhead; On the other hand it may shorten subscription time by registering to a nearer multicast receiver rather than a remote multicast source. As shown in Fig.1, the main ring p, q, r, e, d, o is the first level; branch o, c, x and assist ring p, q, r, f, h, i, k, l are the second level. To achieve hier-

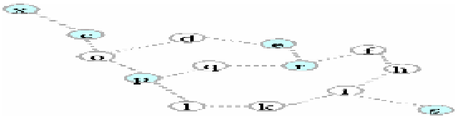


Fig. 1. Hierarchical Eulerian ring-oriented topology

archy we employ master and diverge node. Each ring and branch would have a master. Master of main ring is honored to manage its members including diverge nodes; the diverge node stores information of assist ring or branch in lower level; assist ring master play the same role in their own rings as main ring master. Both master and diverge node are the basis of hierarchical architecture. Note, each node belongs to one but only master from whom it gets multicast packets. Diverge node locating at the joint point has dual duty, one is to do the work of his own ring such as forwarding multicast packets, relaying token in appropriate direction and reporting error message; the other is to establish hierarchical level by transferring multicast packets to lower level.

1.3 Master Election

The algorithm of electing master should be carefully considered because it is the master's duty to perform multicast task and ring recovery. In order to simplify our model, we suppose each mobile node has same computation capability and choose only 2 parameters for master election: average error packets rate and battery capacity. The former is a kind of BER, and proposed by [3]; the latter emphasizes the lifetime of power, and it is proposed by [4]. $New_{P_k^e[i,j]}$ is defined as average error packet within sampling period K for a given direction-oriented wireless link from i to j, $P_k^e[i,j]$ as average error packet after sampling period K for a given direction-oriented wireless link from i to j.

$$Pe^{k}_{[\mathrm{i,j}]} = (\mu \times Pe\,^{k-1}_{[i,j]}) + ((1-\mu) \times New\_Pe^{k}_{[i,j]}\ ) \qquad i,jV,\ 1 > u > 0 \tag{1}$$

LT_i^k is defined as live time estimated after sampling period. LT_i^k is calculated as:

$$LT_{\mathrm{i}}^k = \rho_i^{-1} \times (F_i/E_{i(t)})^{-\alpha} \tag{2}$$

Herein $E_{i(t)}$ means remaining battery capacity of node i at time t; F_i is full-charge battery capacity of node i; α is a positive weighting factor for ratio of the remaining; ρ_i stands for the transmit power at node i at time k. We define Pi as the possibility that a node has to become a master:

$$P_{\mathrm{i}} = \left(LT_i^k\right) \times \left(\sum Pe^{k}_{[i,j]}/N\right) \tag{3}$$

The $\sum \mathrm{P^k_{e\,[i,j]}/N}$ means the average error rate among links processed by node i, and LT_i^k indicates the live time that Ni maybe last after sampling period, the longer live time means the higher possibility to be elected as master.

2 Simulation Result and Conclusion

Mobile hosts are placed randomly within a 1000m*1000m area and moved randomly. Radio propagation range is 250 m. The number of mobile host is 40 and member of multicast group increases from 0 to 30, ranging in the set 0, 5, 10, 15, 20, 25, 30 . The channel capacity is 2Mbits/s and buffer size is 64000bits The X-axis in Fig.2 stands for the number of multicast member (MM). The Y-axis stands for end-to-end delay in average and its unit is ms. End-to-end delay of our proposal increases smoothly, ranging from 1 ms to 2 ms as the number of multicast member increases. In this paper, a novel model of Hierarchical Eulerian Ring-Oriented Multicast Protocol is proposed. A graph based multicast schemes including hierarchical architecture, multicast agent and relative algorithms have been discussed. Simulation results indicate that the proposed scheme outperforms ARMIS, CAMP and ITAMAR in terms of service latency to some extent. We hope our scheme could shed a little light of graphoriented multicast scheme over mobile ad hoc network.

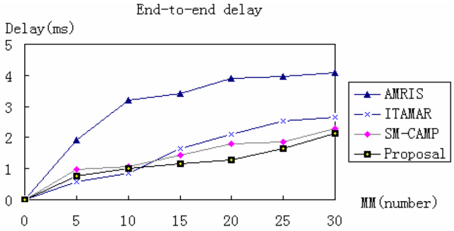


Fig. 2. End-to-end delay

References

1. Carlos de Morais Cordeiro, H. Gossain, and D.P. Agrawal, "Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions, " IEEE Network, Vol.17, No.1, Jan.2003.
2. R.J. Wilson and J.J.Watkins, Graphs: an introductory approach, John Wiley and Sons, Inc Page 122-127.
3. Ying chun, liu yong, shi meilin, "A hybrid protocol in Ad hoc network", 2001 journal of software, China.
4. M. Maleki, K. Dantu and M. Pedram, "Power-aware Source Routing Protocol for Mobile Ad hoc Network", International Symposium on Low Power Electronics and Design, 2002 IEEE.

Performance Analysis of a Multimedia CDMA System Using Dynamic Rate Control

Seung Sik Choi

Computer Eng. University of Incheon,
177 Dohwa-dong Nam-gu Incheon, Korea 402-749,
PH: +82-32-770-8498, FAX: +82-32-766-6894
sschoi@incheon.ac.kr

Abstract. In this paper, we propose an effective access control scheme in an voice/video/data integrated CDMA system. The proposed scheme schedules the transmission of data traffic during low voice/video traffic density period. In addition, to guarantee the delay requirements of data traffic, the transmission rate control of video traffic is triggered when the access delay of data traffic exceeds delay requirements. Numerical results show that outage probability is decreased by allowing the delay of data traffic and decreasing the transmission rates of video traffic.

1 Introduction

Recently, researches on an integrated voice/data CDMA system are performed. In reference [1], the power resource management concept was introduced and the feasible condition was derived. In reference [2], the capacity for data calls can be increased by scheduling data transmissions during low voice load and curtailing data transmissions when the voice traffic is heavy. This control scheme was extended and the prediction control scheme was suggested[3]. To support multimedia traffic in next generation systems, it is required to control resources in reverse links[4]. In reference [5], the transmission rate control of video traffic over wireless channels is proposed. Therefore, the increase of delay sensitive data traffic such as internet web service, requires to control not only data traffic but also video traffic. In this paper, we propose an effective access control scheme using delay and transmission control.

2 Proposed Access Control Scheme

The main characteristics of the proposed scheme is to control the delay of data traffic and the transmission rates of video traffic. Delay-based access control(DAC) scheme enables data traffic to be transmitted during inactive period of voice/video traffic. But, this scheme can't guarantee the quality of service(QoS) of data traffic because data traffic has larger delay in the case of heavy traffic condition. Due to this reason, it is required that video traffic should also be controlled when the delay requirements of data traffic can't be guaranteed. For video

R. Boutaba et al. (Eds.): NETWORKING 2005, LNCS 3462, pp. 1475–1478, 2005.

traffic, rate control scheme changes the transmission rate of video codec. For example, if the access delay of data traffic exceeds the access delay requirement, it triggers the transmission rate control of video traffic. Then, the station decreases the transmission rates of video traffics according to feedback information. In this paper, the access control scheme is called Delay and Transmission-based access control(DTAC) scheme.

3 Performance Analysis

Let the transmission rate of voice/video/data be r_u, r_v and r_w and the E_b/N_o requirement of each class be γ_u, γ_v and γ_w. The number of voice/video/data users are K_u, K_v and K_w, respectively. If mobile stations in each class have the same transmission rate and BER requirement, the feasibility condition in n-th slot can be obtained as

$$S(n) = \frac{U(n)}{a_u} + \frac{V(n)}{a_v} + \frac{W(n)}{a_w} < 1 \tag{1}$$

where

$$a_u = 1 + \frac{B}{r_u \gamma_u}, \; a_v = 1 + \frac{B}{r_v \gamma_v}, \; a_w = 1 + \frac{B}{r_w \gamma_w} \tag{2}$$

Here B is the total bandwidth. Based on the number of packets in the current slot, the permission probability for data traffic in the next slot is computed so that the target outage probability could be met.

Define $R(u, v)$ as the number of data users which are allowed in the next slot based on current active voice and video users. We can obtain $R(u, v)$ based on equation (1), the access probability of data traffic in next (n+1)th slot can be calculated as following.

$$p(n+1) = \max \left\{ p \in [0,1] \middle| \sum_{j=0}^{\lfloor R(u,v) \rfloor} \binom{K_w}{j} p^j (1-p)^{K_w - j} = \delta \right\} \tag{3}$$

where 1-δ is the outage probability requirements(normally 0.01). If the delay requirement of data traffic is D_{max} slot, the access probability of data users[4],p_{min}, can be computed by

$$p_{min} = \frac{1}{1 + D_{max}} \tag{4}$$

If $p(n+1)$ is greater than p_{min}, delay-based access control can be used. The minimum number of data users, $N_{w,min}$, is given by

$$N_{w,min} = \min \left\{ k \in (0, K_w) \middle| \sum_{j=0}^{k} \binom{K_w}{j} p_{min}^j (1 - p_{min})^{K_w - j} = \delta \right\} \tag{5}$$

Let us define a'_v as the modified value of a_v, when the transmission rate control is applied. From (1), a'_v can be given by

$$a'_v > \frac{V(n)}{1 - \dfrac{U(n)}{a_u} - \dfrac{N_{w,min}}{a_w}} \tag{6}$$

The transmission ratio of video traffic, $q(n+1)$, where $0 \le q \le 1(1 :$ normal transmission, $0 :$ no transmission), is computed from (2) as following

$$q(n+1) = \begin{cases} \dfrac{B}{r_v \gamma_v (a'_v - 1)}, & if \quad p(n+1) < p_{min} \\ 1, & otherwise \end{cases} \tag{7}$$

If the access probability of data traffic, $p(n{+}1)$, is lower than minimum value,p_{min}, video transmission control is triggered.

4 Numerical Results

Consider a CDMA system with three classes of service such as voice, video and data. Each traffic has ON/OFF characteristics. The Eb/No, which is defined for service characteristics of voice, video and data traffic, is 6, 7 and 8 dB respectively. The system bandwidth is 32Mhz and minimum access delay in DTAC scheme is constrained by p_{min}=0.5. Based on these parameters, performances of DTAC scheme proposed in this paper is compared with those of DAC(delay-based access control) scheme and no access control scheme. The

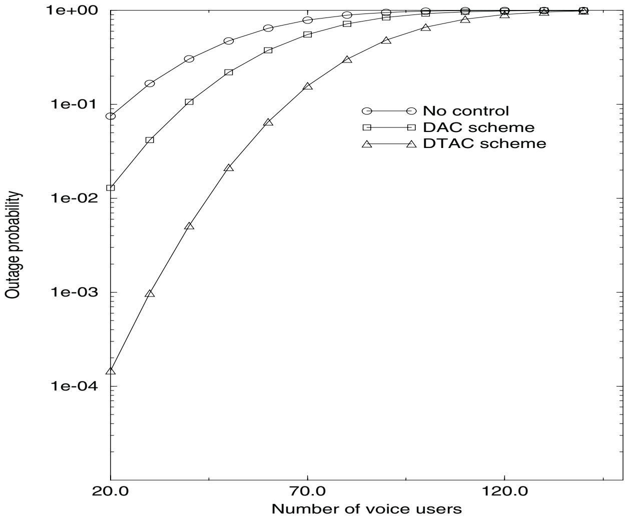


Fig. 1. Outage probability vs. number of voice users(when K_v=7, K_w=10 and Eb/No of data = 8 dB)

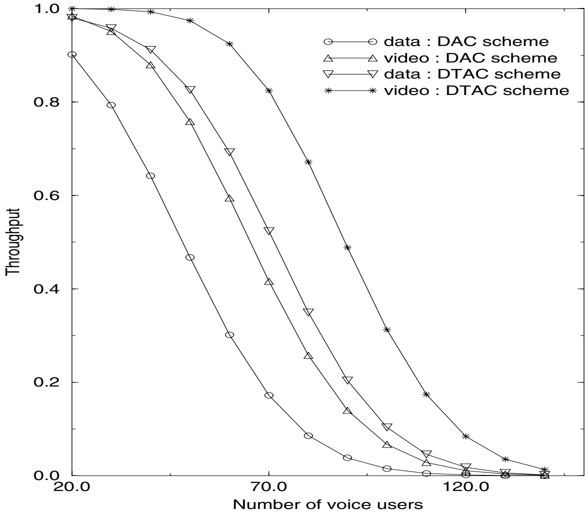


Fig. 2. Throughput vs.number of voice users(when K_v=7, K_w=10 and Eb/No of data = 8 dB)

outage probability of three control schemes is shown in Fig. 1. It is shown that DTAC scheme has lower outage probability compared with two other schemes because it controls access of data and video traffics according to the traffic condition. The data/video throughput is shown in Fig.2. The data throughput in DTAC scheme is increased because the outage probability is decreased by the transmission rate control of video traffic. It is noted that the video throughput of DTAC scheme is increased even though the transmission rates of video traffics is decreased in heavy traffic condition. This result is due to the fact that the outage probability is decreased by DTAC scheme. It is shown that we could get more advantages by decreasing transmission rates of video traffics.

References

1. A.Sampath, P.S.Kumar, and J.M. Holtzman, ”Power control and resource management for a multimedia wireless CDMA system,” in proc. PIMRC’95, Toronto, Canada, Sept. 1995.
2. N.B. Mandayam and J.M. Holtzman,”Analysis of a simple protocol for short message data service in an integrated voice/data CDMA system,” in Proc. IEEE MILCOM’95, San Diego, CA, Nov. 1995.
3. Ashwin Sampath, Jack M. Holtzman, ”Access Control of Data in Integrated Voice/Data CDMA Systems : Benefits and Tradeoffs,” IEEE Journal of Selected Areas in Communications, Vol.15, No.8, pp.1511-1526, October 1997.
4. V.Huang and W.Zhuang, ”QoS-Oriented Access Control for 4G Mobile Multimedia CDMA Communications,” IEEE Communication Magazine, pp.118-125, March 2002.
5. Chi-Yuan Hsu, Antonio Ortega ,and M. Khansari ”Rate Control for robust video transmission over burst-error wireless channels,” IEEE Journal of Selected Areas in Communications, Vol.17, No.5, pp.756-773, May 1999.

Author Index

Abendroth, Dirk 561
Abichar, Zakhia G. 358
Afifi, Hossam 277
Agoulmine, Nazim 277
Aksu, Aylin 828
Al-Duwairi, Basheer 191
Alemu, Tigist 1146
Almeroth, Kevin C. 1180
Alnuweiri, Hussein 90
Altman, Eitan 486, 610
Amaral, José Nelson 574
Ammar, Mostafa 943
An, Sun-Shin 1370
Anelli, Pascal 1108
Anjum, Farroq 1069
Arya, Vijay 992, 1430
Assi, Chadi 792

Banerjee, Sujata 930
Banerjee, Suman 1005
Banner, Ron 536
Bare, Abhijit A. 78
Barman, D. 610
Bechler, Marc 869
Belding-Royer, Elizabeth M. 1180
Benbadis, Farid 1156
Benslimane, A. 610
Bergamini, Andrea 523
Bernard, Laurent 277
Berube, Paul 574
Bettati, Riccardo 586
Beylot, Andre-Luc 1231
Bíró, József J. 718
Black, Richard 252
Blondia, C. 342
Bonaventure, Olivier 1357
Bos, Herbert 743
Boschi, Elisa 1397
Bouabdallah, Nizar 1231
Boutaba, Raouf 1309
Braun, Torsten 1430
Brustoloni, José Carlos 154
Brynjúlfsson, Björn 264
Bulbul, Kerem 828
Butt, Ali R. 1

Cai, Lin 840
Calas, Yvan 1146
Cardei, Mihaela 1393
Castelluccia, Claude 1083
Cen, Zhiwei 1409
Chae, Dong-Hyun 1370
Chaing Chua, Kee 598
Chang, Hoon 1330
Chang, J. Morris 358
Chanson, Samuel T. 980
Charzinski, Joachim 1056
Chatranon, Gwyn 930
Chen, K. 419
Chen, Ling-Jyh 780
Chen, Meng Chang 382
Chhabra, Parminder 730
Chiasserini, C.-F. 1343
Cho, Choong-Ho 1255
Cho, Jinsung 968
Cho, Soohyun 586
Cho, Yongwoo 968
Chobanyan, Alexander 316
Choi, Changho 178
Choi, Dong You 1353
Choi, Seung Sik 1475
Chong Wong, Tung 598
Chrisment, Isabelle 138
Chrobak, M. 328
Chuah, Chen-Nee 1348
Cristea, Mihai Lucian 743
Cui, Jun-Hong 328, 906

Dai, Fei 499, 1393
Dán, György 684
D'Antonio, Salvatore 1397
Davik, Fredrik 1461
de Amorim, Marcelo Dias 1156
Deb, Supratim 1321
de Bruijn, Willem 743
de Launois, Cédric 1357
Deogun, Jitender S. 1449

Dianati, Mehrdad 1418
Dong, Yingfei 178
Donnelly, Austin 252
Dou, Whenhua 53
Dovrolis, Constantinos 647
Dutta, Rudra 1440

El Azouzi, R. 610
El Khayat, Ibtissam 549
Ercetin, Ozgur 828
Eriksson, Jakob 1206
Ertaul, Levent 102

Faloutsos, Michalis 328, 853, 1206
Fawaz, W. 419
Fdida, Serge 1108, 1156
Fei, Zongming 881
Festor, Olivier 138
Fladenmuller, Anne 511
Fodor, Gábor 205
Fodor, Viktória 684
Foong, Annie 65
Friedman, Timur 659
Fu, Xiaoming 289
Fuhrmann, Thomas 1366

Ganguly, S. 1043
Gao, Lixin 1334
Gao, Ruomei 647
Garcia-Luna-Aceves, J.J. 1096
Garetto, M. 1343
Gaudet, Vincent C. 574
Gavrilescu, Alexandru 252
Gburzyński, Pawel 1457
Gebotys, Catherine 126
Gerla, Mario 780, 906
Geurts, Pierre 549
Ghaderi, Majid 1309
Ghamri-Doudane, Yacine M. 277
Ghosal, Dipak 240
Girod-Genet, Marc 277
Gjessing, Stein 1461
Guérin, Roch 634
Guillemin, Fabrice 918
Guitton, Alexandre 1405
Gulyás, András 718
Gurbuz, Ozgur 828
Günther, André 768
Gwanyeon, Kim 370

Haddadou, Kamel 277
Hagsand, Olof 1339
Han, Kyeong-Eun 1426
Han, Kyu-Ho 1370
Hao, B. 1043
Harmantzis, Fotios C. 1445
Harras, Khaled A. 1180
Hassanein, Hossam S. 1267, 1422
Hausheer, David 40
Hayel, Yezekael 816
He, Yang 1426
Helgason, Ólafur Ragnar 264
Heszberger, Zalán 718
Hidell, Markus 1339
Hjálmtýsson, Gísli 264
Hoene, Christian 768
Hogrefe, Dieter 289
Hong, Choong Seon 1471
Hu, Y. Charlie 1
Hui, Joseph Y. 1374
Huo, Wei 792
Huzur, Saran 730
Hwang, Geng-Dian 1413

Ip, Alan T.S. 1383

Jaap, Sven 869
Jayasumana, Anura P. 78
Jean-Marie, Alain 1146
Jha, Sanjay 1005
Jiang, Yixin 114
John, Ajita 730
Joung, Yuh-Jzer 1413
Jun, Kyungkoo 1379
Jun, Sungik 370
Jungmaier, Andreas 1453

Kaheel, Ayman 90
Kalyanaraman, Shivkumar 992
Kanchana, Jayashree 154
Kang, Kyungtae 968
Kang, Seokhoon 1379
Karbhari, Pradnya 943
Karlsson, Gunnar 684
Kasnavi, Soraya 574
Katrinis, Kostas 1326
Kencl, Lukas 523
Kenward, Gary W. 1309
Kherani, Arzad A. 486

Ki, Young Min 623
Kiba, Jean-Etienne 1435
Kim, Dong Ku 623
Kim, Eun Sun 623
Kim, Gwanyeon 370
Kim, Min Sik 893
Kim, Young-Chon 1426
King, Valerie 956
Kleinrock, Leonard 709
Korkmaz, Turgay 53
Koutsakis, Polychronis 1362
Krishnamurthy, Srikanth V. 853, 1206
Krishnamurthy, V. 328
Krishnamurthy, Vikram 1243

Labrador, Miguel A. 930
Lam, Simon S. 893
Lambert, J. 342
Lao, L. 328
Lao, Li 906
Laoutaris, Nikolaos 455
Latapy, Matthieu 659
Lau, William 1005
Law, Lap Kong 853
Leckie, Christopher 1133
Leduc, Guy 549
Lee, Hyong-Woo 1255
Lee, Mahnhoon 1401
Lee, Minsoo 370
Lee, Sanghwan 27
Lee, Seung-Hyun 1426
Leguay, Jérémie 659
Leung, Kin K. 395
Leung, Victor C.M. 1255
Li, Guangzhi 804
Li, Haizhon 395
Li, Yi 893
Li, Zhi 1348
Liang, Ben 1193
Liang, Weifa 431
Lim, Kyung-Soo 1370
Lin, Chuang 114
Ling, Xinhua 1418
Liu, Chong 956
Liu, Jiangchuan 1383
Lochin, Emmanuel 1108
Lu, Weimin 102
Lui, John C.S. 1383

Mahdy, Ahmed M. 1449
Malone, Paul 1397
Mandjes, Michel 303, 561
Mandrekar, V.S. 316
Manimaran, G. 191
Mano, C.D. 166
Mark, Jon W. 598, 840
Marshall, Alan 1031
Martignon, F. 419
Martin, Ruediger 1056
Maxemchuk, Nicholas F. 1168
May, Martin 1326
Mazumdar, Ravi 918
McAlpine, Gary 65
Médard, Muriel 1321
Meddahi, Ahmed 277
Menth, Michael 1056
Mišić, Jelena 228
Mišić, Vojislav B. 228
Michiardi, Pietro 486
Minturn, David 65
Misra, Vishal 1330
Mitrou, N. 1297
Mohapatra, Prasant 240, 1348
Molva, Refik 486
Moulierac, Joanna 1405
Muezerie, André 1457
Mukherjee, Biswanath 1426
Murthy, S. 1043
Mutka, Matt W. 316, 1409

Nah, Jaehoon 370
Naik, Sagar 126, 1418
Nandy, Biswajit 469
Narayan, Sandhya 240
Nasser, Nidal 1281
Ni, Qiang 395
Nikolaidis, Ioanis 1457

O'Neill, Shane 1031
Orda, Ariel 536

Pan, Jianping 840
Pandya, Julee 240
Park, Sehyun 370
Paul, Olivier 1435
Peng, Tao 1133
Percus, A.G. 328
Perros, Harry G. 697
Piratla, Nischal M. 78

Plattner, Bernhard 1326
Potika, Katerina 1465
Psychis, Spyros 1362
Pujolle, Guy 419, 1231
Puttasubbappa, Vishwas S. 697

Ramamohanarao, Kotagiri 1133
Rangarajan, Hari 1096
Rathgeb, Erwin P. 756, 1453
Regnier, Greg 65
Rejaie, Reza 15
Ridoux, Julien 511
Riebach, Stephan 756
Rosenberg, Catherine 918
Rouskas, George N. 406

Safwat, Ahmed M. 1267
Salah Bouassida, Mohamed 138
Salamatian, Kavé 511, 659
Saleh, Aladdin O. M. 1193
Saletore, Vikram 65
Salyers, David 1219
Sanadidi, M.Y. 780
Sarakis, L. 1297
Sarkar, Saswati 1069
Saran, Huzur 730
Saxena, Nitesh 1083
Schmoll, Carsten 1397
Sen, A. 1043
Shafi, Shairmina 228
Shami, Abdallah 792
Shen, B.H. 1043
Shen, Xuemin 114, 840, 1418
Shi, Minghui 114
Shih, Wei-Kuan 382
Shin, Heonshik 968
Sichitiu, Mihail L. 1440
Siddiqi, Amir 469
Sjödin, Peter 1339
Song, Ohyoung 370
Sridharan, Ashwin 634
Srinivasan, Sridhar 1018
Starobinski, David 443
Stathopoulos, P. 1297
Stavrakakis, Ioannis 455
Stiller, Burkhard 40
Striegel, Aaron 166, 1219
Stutzbach, Daniel 15
Subhadrabandhu, Dhanant 1069
Sudeep, Sandeep 1374

Sun, Tony 780
Sun, Yeali S. 382
Sundararajan, Jay Kumar 1321

Takahara, Glen 1422
Tam, Y. Hung 1267
Telek, Miklós 205
Telelis, Orestis 455
Tewari, Saurabh 709
Teymori, Shahram 217
Thaler, Dave 252
Toedtmann, Birger 756
Tuffin, Bruno 816
Tung, Li-Ping 382
Turletti, Thierry 992, 1430

Uhlig, Steve 1357

van de Meent, Remco 303
van den Berg, Hans 561
Van Houdt, B. 342
Vanwormhoudt, Gilles 277
Viniotis, Yannis 511

Walkowiak, Krzysztof 1120
Wan, Yingyu 431
Wang, Dongmei 804
Wang, Feng 1334
Wang, Jun 1449
Wang, Mei 671
Wang, Quanhong 1422
Wei, Jianbin 1388
Willke, Theodore L. 1168
Wolf, Lars 869
Won, Jeong-Jae 1255
Woo, Sung Il 623
Woo, Tim 126
Woods, Roger 1031
Wu, Jie 499, 1393
Wu, Kui 395, 956
Wu, Tao 443

Xi, Ning 316, 1409
Xia, Haidong 154
Xiao, Yang 395
Xu, Cheng-Zhong 1388
Xu, Kenan 1422

Yaipairoj, Saravut 1445
Yan, Jinyao 1326
Yang, Guang 780

Yang, Li 406
Yang, Mengkun 881
Yang, Shuhui 1393
Yang, Yubai 1471
Yi, Jeong Hyun 1083
Ying, Yu 918
Yoon, Yongik 1401
Yu, Fei 1243
Yu, Yinzhe 27
Zahran, Ahmed H. 1193
Zappala, Daniel 15
Zegura, Ellen W. 647, 943, 1018
Zhang, Fan 980
Zhang, Rongmei 1
Zhang, Zhi-Li 27, 178
Zheng, Yanxing 53
Zhu, Danyu 1409
Zhuang, Weihua 217
Zissimopoulos, Vassilios 455