Lecture Notes in Computer Science 2156
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Mikhail I. Smirnov Jon Crowcroft
James Roberts Fernando Boavida (Eds.)

Quality of Future Internet Services

Second COST 263 International Workshop, QofIS 2001
Coimbra, Portugal, September 24-26, 2001
Proceedings

Springer

Volume Editors

Mikhail I. Smirnov
GMD FOKUS
Kaiserin-Augusta Allee 31, 10589 Berlin, Germany
E-mail: smirnov@fokus.gmd.de

Jon Crowcroft
University College London, Department of Computer Science
Gower Street, London WC1E 6BT, UK
E-mail: jon.crowcroft@cs.ucl.ac.uk

James Roberts
France Telecom R&D
38 rue de Général Leclerc, 92794 Issy-Moulineaux, Cedex 9, France
E-mail: james.roberts@francetelecom.com

Fernando Boavida
Universidade de Coimbra, Departamento de Engenharia Informática
Pólo II, 3030-290 Coimbra, Portugal
E-mail: boavida@dei.uc.pt

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Quality of future Internet services : second COST 263 international workshop ; proceedings / QofIS 2001, Coimbra, Portugal, September 24 - 26, 2001. Mikhail I. Smirnov ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2156)
ISBN 3-540-42602-7

CR Subject Classification (1998): C.2, H.4, H.3, J.1

ISSN 0302-9743
ISBN 3-540-42602-7 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Steingräber Satztechnik GmbH
Printed on acid-free paper SPIN: 10845559 06/3142 5 4 3 2 1 0

Preface

The 2001 International Workshop on Quality of future Internet Services (QofIS□2001) held in Coimbra, Portugal, organized by COST Action 263, is the second of what we expect will become a series of successful QofIS workshops. The previous workshop was held in Berlin in the year 2000.

The areas of interest of QofIS cover the design, implementation and provision of Quality of Service, spanning key issues of current and emerging communication systems such as packet-level issues, flow-level issues, network-level issues, architectural issues, and applications.

The emphasis of the QofIS□2001 workshop is on *horizontal* (end-to-end) as well as *vertical* (top-down) provision of quality of services, covering all components of end systems and networks, with the aim of identifying solutions enabling feasible and coherent QoS provision.

The QofIS□2001 call for papers attracted 87 submissions from 23 Asian, Australian, European, North American, and South American countries. These were subject to thorough review work by the Programme Committee members and additional reviewers who carried out their work using a specially built conference system, WebChairing, developed in Coimbra by Flor de Utopia, that allowed full control of the submission and reviewing processes. Based on the comments and recommendations of the reviews, the final programme was defined in a Programme Committee meeting held at the University of Karlsruhe on June 5th, 2001.

A high-quality selection of 22 full papers organised in six, single-track sessions made up the QofIS□2001 main technical programme, which included topics such as QoS support for multimedia communication, admission control, QoS routing, Differentiated Services networks, QoS monitoring and mapping, and traffic engineering. This was complemented by two invited keynote talks: Ian F. Akyildiz from Georgia Institute of Technology, Atlanta, USA opened the workshop with a presentation on problems and research issues of an IP QoS physical testbed setup in collaboration with NASA Qbed to support MPLS traffic engineering and Differentiated Services. Jens Zander, from the Royal Institute of Technology, Sweden, addressed key problems and possible solutions for affordable quality of service guarantees in future wireless multimedia systems in the keynote talk of the second day.

In addition to the main technical programme, the third day of the workshop was dedicated to a mini-track on QoS charging organized by M3I and to a session of invited papers on various European projects/actions: Cadenus, Tequila, Aquila, COST Action 264, and Sequin/GEANT.

We wish to record our appreciation of the efforts of many people in bringing about the QofIS□2001 workshop: to all the authors that submitted their papers to the workshop, regretting that it was not possible to accept more papers; to the Program Committee and to all associated reviewers; to our sponsors and supporting institutions; and to all the members of COST Action 263 who, again, successfully organised a QofIS workshop; to ESEC for the multimedia coverage of the workshop. Finally, we would like to thank all the people that helped us at the University of

Coimbra, namely Jo□o Orvalho, Jorge S□ Silva, Gon□alo Quadros, Paulo Simões, Mar□lia Oliveira, Lu□s Alexandre Silva, Jo□o Cunha, Jo□o S□ Marta, and all the volunteers from the Laboratory of Communications and Telematics.

July 2001

Mikhail Smirnov
Jon Crowcroft
Jim Roberts
Fernando Boavida

Organisation

QofIS2001 was organised by COST *Cooperation europeenne dans le domaine de la recherche scientifique et technique*, Action 263 Quality of future Internet Services.

Steering Committee

Michael Smirnov, GMD FOKUS, Germany (Chairman)
Jon Crowcroft, University College London, UK
James Roberts, France Telecom R&D, France
Fernando Boavida, University of Coimbra, Portugal

Program Committee

F. Boavida, Univ. Coimbra, Portugal (Chairman)

A. Azcorra,, UC3M, Spain
F. Baker, Cisco Systems, USA
J. L. van den Berg, KPN Research, NL
C. Blondia, UIA, Belgium
O. Bonaventure, FUNDP, Belgium
G. Carle, GMD FOKUS, Germany
O. Casals, UPC, Spain
J. Crowcroft, UCL, UK
H. de Meer, UCL, UK
P. de Sousa, IST, EU
J. Domingo-Pascual, UPC, Spain
H. Esaki, Tokyo University, Japan
D. Hutchison, Univ. Lancaster, UK
R. Jain, Nayna Networks/OSU, USA
J. Jormakka, HUT, Finland
G. Karlsson, KTH, Sweden
K. Kilkki, Nokia, USA
P. Kuehn, IND, Germany
M. Luoma, HUT, Finland
E. Madeira, UNICAMP, Brazil
R. D. van der Mei, KPN Research, NL
E. Monteiro, Univ. Coimbra, Portugal
G. Pavlou, Uni Surrey, UK
M. Popa, Procetel, Romania
J. Roberts, CNET, France
C. Scoglio, Georgia Inst. of Tech., USA
D. Serpanos, University of Patras, Greece
M. Smirnov, GMD FOKUS, Germany
J. Sol-Pareta, UPC, Spain
I. Stavrakakis, UOA, Greece
H. Stuettgen, NEC, Germany
G. Ventre, UNINA, Italy
J. Virtamo, HUT, Finland
L. Wolf, Univ. Karlsruhe, Germany
A. Wolisz, TU-Berlin, Germany
J. Wroclawski, MIT, USA
A. Zehl, T-Nova, Germany

Reviewers

S. Aalto, Helsinki Univ. Tech., Finland
B. Ahlgren, SICS, Sweden
T. Anjali, Georgia Tech., USA
M. Arienzo, Univ. Naples, Italy
J. M. Barcelo, UPC, Spain
M. Bechler, Univ. Karlsruhe, Germany
G. Bolch, Univ. Erlangen, Germany
M. Calderon, Univ. Carlos III, Spain
R. Canonico, Univ. Napoli, Italy
D. Careglio, UPC, Spain
L. Cerd□, UPC, Spain
X. Costa, NEC, Germany
C. Demiroglu, Georgia Tech, USA
A. Durresi, Ohio State University, USA
J. Garcia Vidal, UPC, Spain
A. Goulart, Georgia Tech, USA
B. Grn□vall, SICS, Sweden
H. Hartenstein, NEC, Germany
M. Ilvesm□ki, HUT, Finland
D. Larrabeiti, Univ. Carlos III, Spain
N. Laoutaris, Univ. Athens, Greece
O. Lepe, DAC/UPC, Spain
J. Mangues-Bafalluy, UPC, Spain
I. Marsh, KTH Stockholm, Sweden
A. Martinez, Univ. Carlos III, Spain
S. Mascolo, Univ. Bari, Italy
X. Masip, UPC, Spain
P. Mendes, Columbia University, USA
C. Nogu□s, UPC, Spain
M. S. Nunes, INESC / INOV, Portugal
J. L. Oliveira, Univ. Aveiro, Portugal
J. Oliveira, Georgia Tech, USA
M. Oliveira, Univ. Coimbra, Portugal
J. Orvalho, DEI CISUC, Portugal
A. Pereira, IPL, Portugal
G. Quadros, Univ. Coimbra, Portugal
J. Quittek, NEC, Germany
S. Romano, Univ. Napoli, Italy
R. Roth, GMD FOKUS, Germany
J. S□Silva, Univ. Coimbra, Portugal
A. Santos, Univ. Minho, Portugal
A. Schrader, NEC, Germany
D. Sisalem, GMD FOKUS, Germany
U. Steve, Univ. Namur, Belgium
P. Trimintzios, Univ. Surrey, UK
R. Valadas, Univ. Aveiro, Portugal
F. Viktoria, KTH IMIT, Sweden
L. Wood, University of Surrey, UK

Local Organising Committee

G. Quadros, Univ. Coimbra, Portugal
P. Simões, Univ. Coimbra, Portugal
J. S□Silva, Univ. Coimbra, Portugal
Jo□o Orvalho, CISUC, Portugal
Mar□lia Oliveira, CISUC, Portugal

Supporting and Sponsoring Institutions

Siemens, Portugal
IPN □Pedro Nunes Institute
Critical Software, Inc.
University of Coimbra, Portugal
Cisco Systems, Portugal
IPC □Polytechnic Institute of Coimbra
IST □Information Society Technologies
Portuguese Foundation for Science and Technology

Table of Contents

Design of Optimal Playout Schedulers for Packet Video Receivers*

Nikolaos Laoutaris, George Boukeas, and Ioannis Stavrakakis

Department of Informatics, University of Athens, 15784 Athens, Greece
{laoutaris,boukeas,istavrak}@di.uoa.gr

Abstract. One way to reduce, or avoid, the loss of intrastream synchronization due to the delay variability introduced by best-effort networks, is by employing application layer buffering and scheduling at a Packet Video Receiver (PVR), resulting in a higher end-to-end delay. In this paper an analytical model is presented that captures the essential trade-off between stream continuity and stream latency. Unlike past related work, stream continuity is not expressed as the accumulated amount of synchronization loss, but as a combination of the accumulated amount, and the variation of the duration of synchronization loss occurrences. This approach allows for a fine grained optimization of stream continuity which has the potential of providing an improved perceptual quality. It is shown that the minimization of the accumulated amount of synchronization loss, and the minimization of the variance of the duration of synchronization loss occurrences, are two competing objectives; the minimization of variance is desirable because it leads to the concealment of discontinuities. The aforementioned presentation quality metrics are considered by the optimal playout policy, which is derived by means of Markov decision theory and linear programming.

1 Introduction

In recent years multimedia services such as internet telephony, desktop videoconference, and video on demand (VOD), have found a place next to traditional data applications like telnet, ftp or the world wide web. These new services require high transmission reliability and stringent end-to-end delay and delay jitter, to be able to maintain intrastream synchronization between successive media units. The networking community is currently focused on developing mechanisms that will enhance the Quality of Service capabilities of best effort network [1,2]. The main effort in providing current best effort networks with QoS mechanism has been undertaken by the IETF which is standardizing the Integrated Services and Differentiated Services architectures. Nevertheless, it is realized that the deployment of new protocols will be a slow process, so much effort is being put

* This work and its dissemination efforts have been supported in part by the General Secretariat for Research and Technology of Greece under grants 99PENED-92 and 99PENED-193 and the IST Program of the European Union under contract IST-1999-10160 (VideoGateway).

in coping with current network limitations by incorporating intelligent adaptive algorithms at the application layer.

Adaptive rate applications fall into two general categories depending on which end of the communicating parties is adapting its rate or buffering capacity. In source rate adaptation [3,4], it is the sending system that adapts to the time-varying bandwidth availability by regulating the rate of its output video stream. On the other hand, Packet Video Receiving systems (PVR's), may buffer some frames in a playout buffer or even make small adjustments in their playout rate in an effort to conceal the effects of jitter (lack of a frame to display caused by excessively delayed frames).

All PVR's buffer incoming frames as a measure against network jitter. Buffering frames in the playout buffer increases the end-to-end latency at the end-user level. The intrastream synchronization improvement that is gained with the addition of the playout buffer is bounded due to the need to keep the buffering delay below a threshold that is specified by the available end-to-end delay budget.

Different applications tolerate different maximum end-to-end latencies. Bidirectional applications such as desktop video conferencing place very strict latency requirements, typically of few hundreds of milliseconds. On the other hand, unidirectional applications, for example video on demand (VOD), allow for much larger latencies in the order of seconds. In a VOD application, the PVR can buffer massive numbers of frames, thus ensure an almost pauseless video presentation across the widest range of network jitter. The absence of critical latency requirements also allows for the incorporation of techniques such as data proxy-ing and client-server feedback which can help in using network resources more efficiently, especially in the case of Variable Bit Rate (VBR) encoded video [5,6,7].

The hard real time requirements of interactive applications [8] are met by the *absolute delay method* which delivers frames at a constant end-to-end latency and drops late frames. Applications with looser delay constraints (soft real time) may present a late frame and thus gain in stream continuity by not discarding a frame which has already harmed the continuity of the stream by causing an underflow with its late arrival. Keeping the late frame increases the end-to-end latency of all subsequent frames resulting in a playout policy with variable overall latency for different frames. Of course, even soft real time applications have an upper bound on latency or a limited buffer capacity so eventually the increase of buffering delay will lead to frame droppings.

The family of playout schedulers that we study in this work does not guarantee a constant end-to-end latency nor a constant buffering delay. The scheduler guarantees only a statistically constant (mean value) buffering delay and is thus more suitable for soft real time applications. The gain from the relaxation of the constant latency requirement, is the ability to react better to bursty frame-arrival sequences forming due to network jitter. Perceptual quality can be improved by implementing *mild* latency control methods, which harm stream continuity less than the *harsh* deadline discard of late frames under the absolute delay method. At the same time, the buffering delay is not ignored – in favor of stream continuity – but is kept statistically constant, below an acceptable, user-defined level. The scheduler increases the buffer occupancy as a measure against jitter, and decreases the buffer occupancy, in order to control the buffering delay and avoid overflows. The playout buffer occupancy is controlled by regulating the playout duration of frames in a per frame basis. This necessity is not present in packet audio systems, where the existence of silence periods gives the system the ability to change the size of the de-jitter buffer by modifying the duration of the silence periods, in a per talkspurt basis, without modifying the duration of media units [9,10,11,12,13,14].

Analytical studies for PVR's that employ dynamic playout schedulers, have recently appeared in the literature. Yuang et al. [15] proposed a dynamic playout policy based on *slowdown* thresholds. In their work, frames are presented at a linearly decreasing rate when the playout buffer occupancy drops below TH – the slowdown threshold – and at a constant rate μ, faster than the mean frame arrival rate λ, when the occupancy exceeds TH. We have modified the threshold-based scheduler of [15] in [16], by looking at the case where the scheduler applies a playout rate that never exceeds the mean frame arrival rate. This modification dictates that the buffering delay will only increase (it decreases when the playout rate exceeds the normal encoding rate) resulting in a scheduler that is more suitable for unidirectional, soft-real-time applications, such as web-based video distribution systems. The design of the scheduler has been simplified by the introduction of a compact and fair continuity metric – the Distortion of Playout (DoP) – which combines all causes of media asynchrony. The study has limited the range of the threshold parameter, TH, by identifying a range of values where there is no beneficial tradeoff between continuity and reduction of mean playout rate – the two antagonistic metrics of interest. Interestingly, it has been shown that this range of values changes with the burstiness of the frame arrival process, revealing the danger of an initially meaningful TH appearing in the undesirable area due to the change of arrival burstiness. The work is supplemented by online algorithms for the detection and the maintenance of the operational parameter TH within the area of beneficial tradeoff across unknown, non-stationary, delay jitter.

With the current work we improve previous heuristic frame-playout policies by formulating an optimization problem that involves the two main metrics of interest; the intrastream synchronization and the buffering delay. Furthermore, the intrastream synchronization metric used in this work is more fair and more general than previously used synchronization metrics [15,16,17,18]. In addition, the buffering delay is controlled in a way that harms stream continuity as little as possible.

The remainder of the paper is organized as follows. Some key concepts and definitions are presented is Section 2. A number of optimal playout adaptation policies for packet video receivers are derived is Section 3 by introducing some interesting metrics and employing Markov decision theory and linear programming techniques. Numerical results are presented in Section 4 together with a comparative analysis with a non-optimal scheduler from the literature. Section 5 comments on implementation issues and future development issues. Section 6 concludes the paper.

2 Definitions

In the following sections we construct a mathematical model for the derivation of the optimal frame-playout policy. Among others, the need for a method to control the duration of frames will arise. Under R_T – the usual static playout policy– all frames are presented with an equal duration T, given by the frame production rate λ, via $T = 1/\lambda$. In some cases, we will need to expand the presentation duration of a frame beyond T, the normal duration. When this decision is repeated for successive frames, it constitutes a transient reduction of playout rate (we will refer to this as a *slowdown*). On the opposite, the presentation of a sequence of frames, with durations that are shorter than T, constitutes a transient increase of playout rate (much like a *fast forward* operation on a VCR). The most general way to regulate B, the duration of a frame, is to allow it to take all non-negative values. This approach gives the utmost freedom in the search for an optimal playout policy. In this paper, however, we limit the possible values for B to a countable set of values that follow:

$$B_k \triangleq k \cdot c$$

where c is an fraction of the normal frame duration T, such that $T = a \cdot c$. The value of a is the *cutting factor* of T. Using the last relationship B_k becomes:

$$B_k = \frac{k}{\lambda \cdot a} \tag{1}$$

$\lambda = 1/T$ is the normal frame rate, typically 25 or 30 frames/sec. c will be the basic unit for shortening and expanding the duration of a frame. The reduction of freedom in the search for optimal policies is negligible for small values of c, e.g., $c = T/10$. In the following, the choice of an appropriate value of k will be referred to as an *action* or a *decision*.

Expanding or shortening the duration of a frame presentation introduces a discontinuity – a loss of intrastream synchronization – quantified by the difference between the selected frame duration and the normal frame duration T. Let d_{ik} denote the *discontinuity* that is incurred when the next frame is presented with a duration B_k and the current buffer occupancy is i.

$$d_{ik} \triangleq |B_k - T| + S \cdot I_{\{i=0\}} \qquad 0 \le i \le N \tag{2}$$

S is a random variable that adds to d_{ik} the effect of buffer underflows. S represents the time interval between a buffer underflow instant and the next arrival instant. I is the indicator function. We define the Distortion of Playout as:

$$DoP_{ik} \triangleq d_{ik} + \bar{l}_{ik} \cdot T \tag{3}$$

where $\bar{l}_{ik}$ [1] is the expected number of lost frames due to buffer overflow over the next presentation interval, given that i is the current buffer occupancy, and decision k is made. $\bar{l}_{ik}$ is given by:

$$\bar{l}_{ik} = \sum_{m=1}^{\infty} m \cdot P\{N - i + 1 + m, B_k\} \tag{4}$$

$P\{x, t\}$ is the probability of x new arrivals occurring in a time interval with duration t. DoP is more fair than d_{ik} as a stream continuity metric, as it also accounts for synchronization losses due to frame overflows.

A basic idea reflected in the definitions of both d_{ik} and DoP_{ik} is that the perceptual cost of an idle time gap between two frames (occurring when the first frame stays on display for more than T) is equal to the perceptual cost of a loss-of-information discontinuity of equal duration. This is based on recent perceptual studies [19] where it is shown that jitter degrades the perceptual quality of video nearly as much as packet loss does [2].

3 Design of Optimal Playout Schedulers

In this section a PVR consisting of a playout buffer and a playout scheduler is studied using Markov decision theory and linear programming. The goal is to derive the optimal playout policy, which by controlling the duration of frames based on the current buffer occupancy, provides a perceptually optimal presentation schedule.

3.1 MDP Problem Formulation

Let $\{I_n\}_{n>0}$ be a stochastic process for i, the number of frames in the playout buffer upon the presentation completion instant[3] of the nth frame. $\{I_n\}$ is a Markov process under the Poisson arrival process which is assumed in this paper for the modeling of frame arrivals. The Poisson process and the associated interarrival-time exponential distribution are much more variable than actual frame arrival processes and thus lead to rather pessimistic performance results. Nevertheless, the memoryless property of the exponential distribution simplifies the model and is thus suitable for a first exposition of the basic ideas that appear in this paper (the implications of the Poisson assumption are further examined in Sect. 5).

To formulate $\{I_n\}$ as a Markov decision process (MDP), we need to define a tuple $\langle \mathcal{S}, \mathcal{A}, P, C \rangle$, where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of possible actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition function specifying the probability $P\{j|i, k\} \equiv p_{ij}(k)$ of observing a transition to state $j \in \mathcal{S}$ after

[1] See [16] for a detailed derivation of $\bar{l}_{ik}$.

[2] For an example in support of this claim; we may think that an underflow with a duration T, degrades stream continuity, nearly as much, as does a lost frame.

[3] Hereafter called an *observation*, or a *decision* instant.

taking action $k \in \mathcal{A}$ in state $i \in \mathcal{S}$ and, finally, $C : \mathcal{S} \times \mathcal{A} \to \Re$ is a function specifying the cost $C(i,k) \equiv c_{ik}$ of taking action $k \in \mathcal{A}$ at state $i \in \mathcal{S}$. A *policy* $R \equiv \{D_{ik} : i \in \mathcal{S}, k \in \mathcal{A}\}$ is a mapping: $\mathcal{S} \times \mathcal{A} \to [0,1]$. A policy is completely defined for a given tuple $\langle \mathcal{S}, \mathcal{A}, P, C \rangle$ by the probabilities

$$D_{ik} \triangleq P\{\text{action} = k | \text{state} = i\} \tag{5}$$

The state space $\mathcal{S}$ of $\{I_n\}$ comprises all possible buffer occupancy levels thus takes values in $[0 \ldots N]$, N being the buffer size. An *action* is defined to be the choice of an integer value k that explicitly determines B_k, the presentation duration for the next frame. The action space for the problem is $\mathcal{A} = [0 \ldots K]$, where K is an integer value that results in the maximum allowable playout duration B_K. For $k = 0$ the playout scheduler discards the next frame.

The action taken at an observation instant affects the evolution of $\{I_n\}$ by affecting the probability law for the next transition. Let P be a $(N+1) \times (N+1)$ matrix containing vectors elements $\boldsymbol{p}_{ij}$, with $K+1$ elements per vector. The $(k+1)$th element of vector $\boldsymbol{p}_{ij}$ is the probability of observing a transition from state i to state j when decision $k, k \in \mathcal{A}$, is made. Let $P\{x,t\}$ denote the probability of observing x new frame arrivals in a time interval of duration t. Since $\{I_n\}$ corresponds to the buffer occupancy, the probability of observing a transition to a state j after playing the next frame for time B_k (selecting action k), depends on the number of new frames that will arrive during B_k.

$$p_{ij}(k) = \begin{cases} P\{j, B_k\} & i = 0, 0 \le j < N \\ 1 - \sum_{m=0}^{N-1} P\{m, B_k\} & i = 0, j = N \\ P\{j - i + 1, B_k\} & 0 < i < N, i \le j < N \\ 1 - \sum_{m=0}^{N-i} P\{m, B_k\} & 0 < i \le N, j = N \\ P\{0, B_k\} & 0 < i \le N, j = i - 1 \\ 0 & \text{elsewhere} \end{cases} \tag{6}$$

In this paper $P\{x,t\}$ follows the Poisson distribution with parameter λ, the frame production rate. Using the transition matrix of (6), we can obtain $\pi_i(R)$, the limiting distribution of $\{I_n\}$ for a particular policy R, by solving the stationary equations $\pi(R) = \pi(R) \cdot P(R)$

Let c_{ik} denote the cost incurred when action k is taken when the occupancy process is in state i. The optimal policy R_{opt} is defined to be the policy that minimizes the expected value of c_{ik} over all $i, k \in \mathcal{S} \times \mathcal{A}$. If π_i denotes the limiting probability that process $\{I_n\}$ is in state i, then

$$R_{opt} = \arg\min_R E\{c\} \quad \text{where} \quad E\{c\} = \sum_{i=0}^{N} \sum_{k=0}^{K} c_{ik} \cdot D_{ik} \cdot \pi_i$$

3.2 Cost Assignment

The cost considered here consists of two components: one that captures the induced lack of continuity, and one that captures the induced buffering delay.

The relative weighing between the two cost components, reflects the desired latency-continuity performance compromise that is to be achieved by the optimal policy.

Continuity Cost. This cost component punishes the *lack of continuity* that may arise from a certain action. This lack of continuity may be directly experienced as in the case of a frame presentation with a duration smaller or larger than T. In addition, the continuity cost also accounts for the anticipated lack of continuity that may arise over the chosen presentation interval as consequence of the chosen action. To be precise, the anticipated lack of continuity refers to the possibility of losing frames due to buffer overflow over the chosen presentation interval.

An candidate for the continuity cost is DoP_{ik}, as defined in (3). Setting $c_{ik} = DoP_{ik}$ returns an $E\{DoP\}$-optimal policy; a policy that provides a minimal expected value for the Distortion of Playout. The results of Sect. 4 indicate that R_T, the static deterministic policy with constant presentation durations equal to the frame period, is $E\{DoP\}$-optimal.

The minimization of $E\{DoP\}$ calls for the minimization of the accumulated amount of synchronization loss which is due to: underflow discontinuities, slowdown discontinuities, overflow discontinuities and fast-forward discontinuities. The minimization of $E\{DoP\}$ is a legitimate objective but cannot guarantee perceptual optimality, as it only caters for the minimization of the accumulated loss of synchronization, without paying any attention as to how this loss of synchronization spreads in time. It has been realized that the human perceptual system is more sensitive to a small frequency of long-lasting disruptions than to a higher frequency of sort-lived disruptions [15]. This is due to human perceptual inability to notice small deviations of presentation rate. As a result, a better perceptual quality can be expected by replacing large continuity disruptions (underflows and overflows) with shorter ones (slowdowns and fast-forwards), even when the later lead to a higher value for $E\{DoP\}$. Thus, a playout policy should be allowed to sacrifice an increase of $E\{DoP\}$ if this increase provides for a smoother spacing between synchronization-loss occurrences, thus help in concealing them. We pursue this idea by defining the state-action cost to be

$$c_{ik} \triangleq \beta \cdot DoP_{ik} + (1 - \beta) \cdot DoP_{ik}^2$$

The weighing factor β is a user-defined input that controls the relative importance between the two minimization objectives; the minimization of the expected value of DoP and minimization of the variability of DoP. Setting $\beta = 1$ leads to the minimization of $E\{DoP\}$ without any regard for the variance of DoP. Setting $\beta = 0$ results in the cost $c_{ik} = DoP_{ik}^2$, demanding the minimization of the expected square value of DoP. The minimization of $E\{DoP^2\}$ returns a policy that distributes synchronization losses more smoothly than $E\{DoP\}$-optimal policies do. As will be shown later, the reduction in the variance of DoP comes at the cost of an increase in the expected value of DoP. Values of β that fall between the two extremes (0 and 1) provide various levels of compromise between

$min\{E\{DoP\}\}$ and $min\{E\{DoP^2\}\}$. Due to the fact that the two quantities have different units (here *seconds* and *seconds*^{2}) small values of β must be used if a meaningful tradeoff is to be achieved, otherwise the policy turns absolutely in favor of $min\{E\{DoP\}\}$.

Latency Cost. In addition to the continuity cost a latency cost is jointly considered to allow for the control of the buffering delay.

The buffering delay for an arriving frame depends on the number of frames i found in the buffer upon arrival, the presentation duration associated with these frames, $B_n, n = 1 \dots i$, and the remaining duration of the frame that is currently being presented, X. The expected buffering delay of an arriving frame that finds i frames in the buffer is:

$$W_i = X + \sum_{n=1}^{i} B_n \tag{7}$$

W_i cannot be expressed explicitly, as B_n's follow a distribution which is not known; in fact, this distribution is is derived in the process of obtaining the optimal policy. Nevertheless, it is expected that the mean value of B_n will not differ significantly from the actual frame duration T (since the desirable mean playout rate induced by the optimal policy is close to and cannot exceed $1/T$). Thus, W_i can be approximated by some value in the interval $[i \cdot T, (i+1) \cdot T]$. $\widetilde{W_i} = i \cdot T$ is chosen as the approximation of W_i. Let $\{A_n\}_{n>0}$ denote the buffer occupancy process on frame arrival instants and let $a_i, i = 0 \dots N$ denote the distribution of $\{A_n\}$. Then the expected buffering delay is approximated by:

$$\widetilde{W} = \sum_{i=0}^{N} a_i \cdot \widetilde{W_i} = \sum_{i=0}^{N} a_i \cdot i \cdot T \tag{8}$$

Equation (8) approximates the mean buffering delay by a function of a_i, the distribution of $\{A_n\}$. It applies that $\{A_n\}$ has the same distribution with $\{I_n\}$, that is, $a_i \equiv \pi_i$ (see Sec. 5.3 in [20] or Sec. 8.3 in [21] for details); thus $\widetilde{W}$ becomes a function of π_i, the distribution of $\{I_n\}$. By regulating the mean value of $\{I_n\}$ we also regulate the delay approximation $\widetilde{W}$ since the latter only depends on the distribution π_i. The mean value of $\{I_n\}$, and thus the delay $\widetilde{W}$, can be regulated by introducing a latency cost component to every action. We define the latency cost for taking decision $k, k \in \mathcal{A}$, when the number of frames in buffer (the state) is i to be

$$L_i \triangleq \frac{1}{T \cdot N} \cdot \widetilde{W_i} = \frac{i}{N} \tag{9}$$

The latency cost does not explicitly depend on the chosen playout duration (determined by k), but only implicitly, as the evolution of $\{I_n\}$ depends on the presentation duration of the current frame. The buffer size N together with T normalize the latency cost to unity.

We add the latency cost to the continuity cost introduced in the previous section by attaching to it a weighing factor γ that controls its relative importance over the continuity cost. The final expression for the transition cost is given by

$$c_{ik} \triangleq \beta \cdot DoP_{ik} + (1-\beta) \cdot DoP_{ik}^2 + \gamma \cdot L_i \tag{10}$$

3.3 Linear Programming Formulation

In this section, we employ linear programming (LP) to derive the optimal policy for the MDP of Sect. 3.1. The decision variables of the linear program are x_{ik}'s which denote the joint probability of being at state $i \in \mathcal{S}$ and performing action $k \in \mathcal{A}$:

$$x_{ik} = P\{\text{action} = k \text{ and state} = i\}$$

The x_{ik}'s do not directly stipulate a policy, a way to select an action at a given state. However, their are required by the *Simplex Method*, used for the solution of the LP, as they are more convenient as decision variables than the actual policy D_{ik}. Having calculated x_{ik}'s, the optimal policy is readily determined since it holds

$$D_{ik} = \frac{x_{ik}}{\pi_i} \quad \text{where} \quad \pi_i = \sum_{k=0}^{K} x_{ik}$$

The function z we wish to minimize is the expected cost over the state-action space, whereas the constraints of the problem reflect the structure of the MDP. The complete problem is the following:
minimize

$$z = E\{c\} = \sum_{i=0}^{N} \sum_{k=0}^{K} c_{ik} x_{ik} \tag{11}$$

subject to

$$\forall j: \sum_{k=0}^{K} x_{jk} = \sum_{i=0}^{N} \sum_{k=0}^{K} x_{ik} p_{ij}(k) \tag{12}$$

$$\sum_{i=0}^{N} \sum_{k=0}^{K} x_{ik} = 1 \tag{13}$$

$$\forall i, k: \; x_{ik} \geq 0 \tag{14}$$

The $N+1$ constraints in (12) are the steady-state equations, suggesting that the steady state probability of being in state j equals the sum of probabilities of being in any other state i and performing a transition to j. Constraint (13) and the $N+1 \times K+1$ constraints in (14) stem from the fact that the x_{ik}'s form a probability distribution over the state-action space and must, therefore, be non-negative and sum to 1.

It can be proved [22,23] that for each state $i \in \mathcal{S}$ there is *exactly* one $x_{ik} > 0$. This essentially means that the optimal policy is deterministic i.e., D_{ik}'s are 0 or 1.

4 Results and Discussion

In this section we demonstrate the effectiveness of the MDP formulation by deriving several numerical results for various parameter configurations. All the examples will be based on a reference system with the following parameters: a playout buffer with space for $N = 50$ frames; a normal frame rate $\lambda = 30$ frames/second; a basic time quantum c, equal to a tenth of the normal frame period T (i.e., a=10 in (1)). The two weighing factors, β, γ of (10), will shape different optimal policies for applications with different continuity-latency requirements.

4.1 Optimizing Stream Continuity

The following numerical results demonstrate policies that perceive the intrastream synchronization quality, as a combination of $E\{DoP\}$ and $Var\{DoP\}$, following the arguments of Sect. 3.2. The latency weight γ is set to zero, thus does not affect the shape of the optimal policy. We focus on the role of continuity weight β, that regulates the relative importance between the mean value and the variance of DoP.

Figure 1 depicts the effect of β on the structure of the optimal policy. The x,y surface illustrates the optimal policy, for a specific value of β (on the z-axis). We let β take values in $[0, 0.1]$ with an increment of 0.001. We have already noted that only small values of β provide a meaningful compromise between mean value and variance. This is due to the fact that the state dependent cost components DoP_{ik} and DoP_{ik}^2 have different units. We save a lot of unnecessary optimization runs by not letting β run up to 1. The optimal policy becomes R_T, statical deterministic with presentation duration T regardless of buffer occupancy, soon after $\beta = 0.1$. For small values of β the policy tries to minimize the variance of DoP. In doing so, it presents frames slower when approaching an underflow and faster when approaching an overflow. For the given numerical example, the maximum frame duration is B_{14} and the minimum B_9. It might seem odd that for an occupancy equal to zero, the optimal policy plays the next frame with a normal duration $B_{10} = T$ and not slower as would be expected. This decision is justified by noting that for the special case of an underflow occurrence, prior to the display of the first arriving frame, an underflow interval exists; the policy does not lengthen the duration of this frame because the associated *slowdown* discontinuity d_{0k}, would become very large as it also contains the underflow interval.

Figures 2,3 illustrate the effect of β on the values of $E\{DoP\}$ and $Var\{DoP\}$. It is evident that small values of β favor the reduction of $Var\{DoP\}$ by increasing $E\{DoP\}$, while the opposite holds for large values of β ($\beta > 0.1$), where the optimal policy returns a small value for $E\{DoP\}$ but with a large variance of DoP. Note that both $E\{DoP\}$ and $Var\{DoP\}$ are not continuous but change in steps at different values of β. This is an expected behavior as different values of β produce policies that may differ at some *specific* state-action pairs. The discrete action space A allows only specific values for $E\{DoP\}$ and $Var\{DoP\}$.

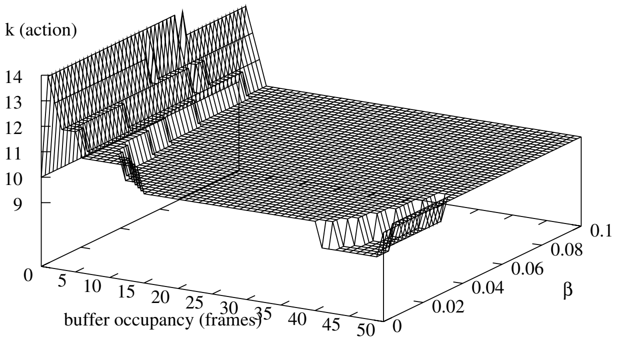


Fig. 1. Optimal policies for various values of β, the weight factor that governs the tradeoff between mean value and variance of DoP. No latency cost is considered here ($\gamma = 0$). The normal frame duration T corresponds to B_{10} i.e., when the decision value k is ten.

Figure 4 reveals a relationship between the continuity weight β, and $E\{I_n\}$, the expected buffer occupancy at the observation instants. Small values of β, cause a slight increase in the occupancy of the buffer. This is justified by looking carefully at the structure of policies that favor the reduction in the variance of DoP (β taking small values). This reduction of variance is achieved by adjusting the duration of frames at the buffer extremes, thus preventing occurrences of variance-increasing events, such as underflows and overflows. Figure 1 shows that presentation slowdowns – occurring when the buffer level drops – are severe (large expansion of the duration of the frames for very low buffer occupancy). In comparison, presentation fast-forwards, are less severe (slight reduction of the duration of frames for very high buffer occupancy). This behavior leads to a slightly reduced mean playout rate which increases the mean buffer occupancy by increasing the limiting probabilities of large occupancy levels.

In the aforementioned analysis, the buffering delay is limited by N, the maximum buffer capacity. It also decreases with β, as shown in Fig. 4. Having neglected γ, the weighing factor of the latency cost, an implicit buffering delay method can be devised, be choosing a limited buffer size N, and use β for more detailed delay adjustments.

4.2 Comparative Analysis with Non-optimal Schedulers

This section demonstrates the performance gains of the optimal scheduler by comparing its performance with the performance of the non-optimal empirical threshold-based playout scheduler of [16] (briefly discussed in Sec. 1). Figure 8 illustrates the performance of the threshold scheduler. The optimal value for the expected value of DoP is achieved for the threshold parameter $TH = 2$ and is $E\{DoP\}_{TH=2} = 6.8 \cdot 10^{-4}$. The variance of DoP for $TH = 2$ is $V\{DoP\}_{TH=2} = 1.69 \cdot 10^{-5}$. Let $X\{DoP\}'_{\beta}$ denote the expected value or the variance of DoP for some value(s) β of the optimal scheduler. The numerical results show that

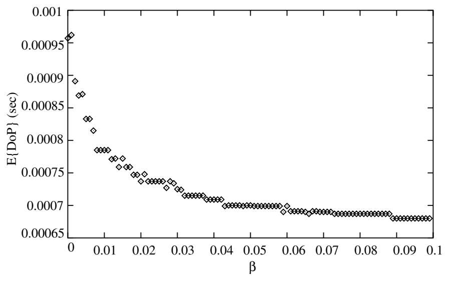


Fig. 2. The effect of the continuity weight β on $E\{DoP\}$. The latency weight γ is set zero.

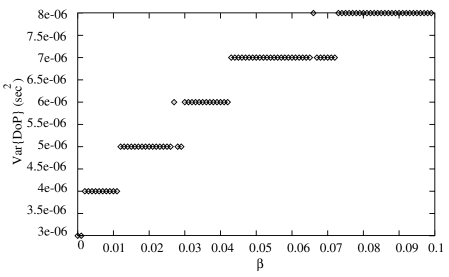


Fig. 3. The effect of the continuity weight β on $Var\{DoP\}$. The latency weight γ is set zero.

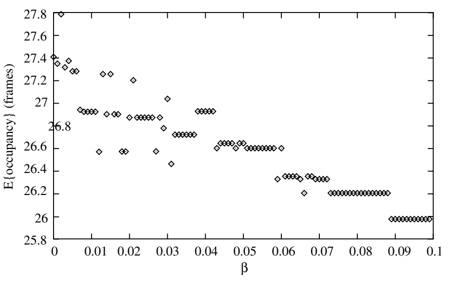


Fig. 4. Expected playout buffer occupancy with β.

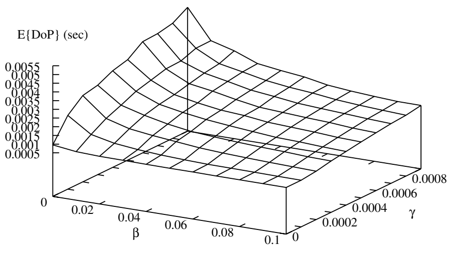


Fig. 5. The effect of β and γ on $E\{DoP\}$.

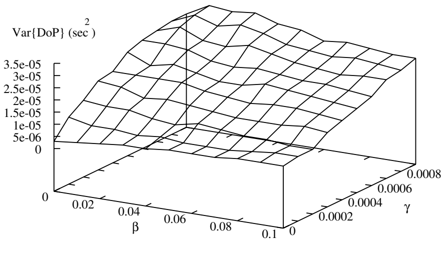


Fig. 6. The effect of β and γ on $Var\{DoP\}$.

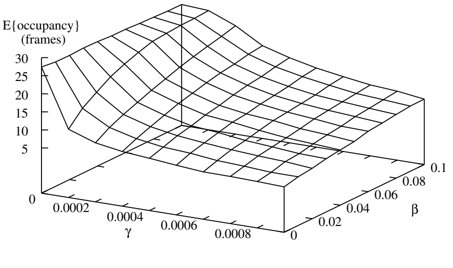


Fig. 7. The effect of β and γ on buffer occupancy.

for the same value of averages, $E\{DoP\}'_{\beta>0.09} = E\{DoP\}_{TH=2}$, the optimal scheduler exhibits twice as better variance of DoP, $V\{DoP\}'_{\beta>0.09} = 0.8 \cdot 10^{-5} < 1.69 \cdot 10^{-5} = V\{DoP\}_{TH=2}$. Fixing the value of the variance at $V\{DoP\}_{TH=14} = V\{DoP\}'_{\beta>0.09} = 0.8 \cdot 10^{-5}$, the optimal policy is again superior by achieving a better expected value of DoP, $E\{DoP\}'_{\beta>0.09} = 6.8 \cdot 10^{-4} < 8.4 \cdot 10^{-4} = E\{DoP\}_{TH=14}$. Note, that for the threshold scheduler, the absolutely smallest variance of DoP is achieved at a threshold value of $TH = 17$: $V\{DoP\}_{TH=17} = 0.79 \cdot 10^{-5}$. We used for the comparison $TH = 14$ instead of $TH = 17$ because for the latter we cannot find a value of β that provides $V\{DoP\}'_{\beta} = V\{DoP\}_{TH=17}$ and fix $V\{DoP\}$ for a numerical comparison, nevertheless, $TH = 14$ is very close to the absolutely smallest $V\{DoP\}$ achieved by the threshold scheduler at $TH = 17$. Also note that the absolutely smallest $V\{DoP\}$, from both methods, is achieved by the optimal scheduler for $\beta = 0$ and it is over two times superior to the optimal $V\{DoP\}$ performance of the threshold scheduler at $TH = 17$. Thus, it is concluded that the optimal scheduler outperforms the threshold-based scheduler in the entire continuum of choices between optimization of $E\{DoP\}$ and optimization of $V\{DoP\}$.

4.3 The Tradeoff between Stream Continuity and Buffering Delay

In this section we show how the latency weight γ of (10) can be used to control the playout buffer occupancy and the associated buffering delay. Our results indicate that the latency control function has a twofold degrading effect on stream continuity; both the expected value of DoP and the variance of DoP, increase with γ. The degradation of synchronization quality has also been exhibited with schedulers that employ *harsh* latency control methods, such as deadline discard of late frames. It also applies to our scheduler, despite the *milder* latency control function, which does not discard frames, but plays them faster in order to control the buffering delay.

Figure 7 illustrates the reduction of the expected value of $\{A_n\}$ with increasing values of γ. For a given latency weight γ, smaller values of β lead to smaller occupancy levels. This happens as the DoP_{ik} costs (adding to c_{ik} more with large βs) are higher than the corresponding DoP^2_{ik} costs (adding to c_{ik} more with small βs); thus larger βs balance more effectively the latency weight γ, which is pushing for greater latency reduction.

By choosing an appropriate latency weight the playout buffer occupancy can be stabilized to a level that adds an acceptable buffering delay component. Figure 5 [4] shows the increase in the expected value of DoP that is paid for the regulation of buffer occupancy. The tradeoff between continuity and buffering delay is apparent.

Figure 6 depicts the effect of the latency weight on the variance of distortion of playout. As we would expect the occupancy control function not only increases $E\{DoP\}$ but also increases $V\{DoP\}$.

5 Implementation Issues and Future Work

It has already been mentioned that the exponential interarrival distribution is much more variable compared to typical interarrival distributions of periodic streams. The optimal policy is expected to approach an optimal performance as long as the variability of a real distribution approaches the variability of the exponential distribution. Such high variability can be realistic in time-windows of

[4] Note that compared to Fig. 7, we have switched the axes for γ and β in order to make the plots more comprehendible.

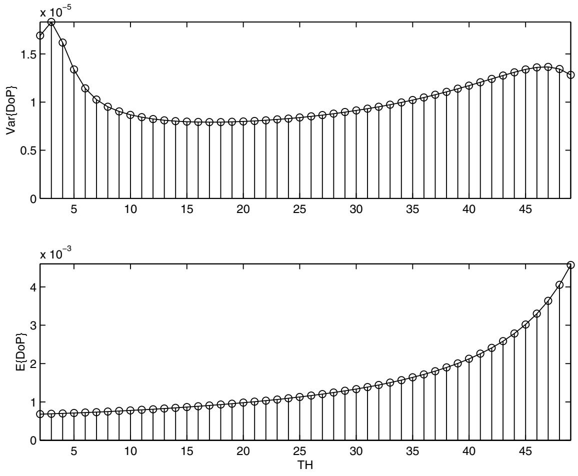


Fig. 8. $E\{DoP\}$ and $V\{DoP\}$ from a threshold-based playout adaptation scheme of [16] where the value of the threshold parameter (TH) regulates the tradeoff between the two metrics. The expected value is minimized for $TH = 2$ and the variance for $TH = 17$. The optimal policy outperforms the threshold-based schedulers in the entire continuum of choices between optimization of $E\{DoP\}$ and optimization of $V\{DoP\}$.

extreme network jitter. However, in times of reduced network jitter the proposed scheme can deviate significantly from the expected optimal performance. To overcome this problem, the analytical model is being expanded to allow for the incorporation of arrival processes that are more regular than Poisson. This will allow for the derivation of different optimal policies according to the current level of network jitter. The envisioned implementation will only need to estimate the current level of network jitter and then *load* the appropriate offline-computed optimal policy. The gain of such an architecture is twofold: the performance of the scheduler is optimized, and the complexity of the system is kept low as no online optimization is performed since offline computed policies are being used.

6 Conclusions

This paper has presented a family of playout schedulers for packet video receivers. The proposed schedulers optimize a meaningful expression of stream quality which involves the two major performance quantities: the stream continuity and the induced buffering delay. A fair and compact stream continuity metric – the Distortion of Playout – has been used as the basis for the detailed assessment of the overall intrastream synchronization quality that caters for both the accumulated amount of discontinuities and variance of their duration. It has been suggested that the joint consideration of the amount and the pattern of synchronization loss has the potential of improving the perceptual quality, compared to the case where only the amount is considered.

The numerical results reveal an important tradeoff that must be considered when designing a packet video receiver. It is realized that the reduction of the accumulated amount of synchronization loss and the attempt to evenly spread it in time, are two antagonistic objectives. Both continuity components are degraded by the introduction of a delay control function, despite the fact that the proposed delay control scheme tries to be friendly towards stream continuity, by avoiding crude operation such as frame discardings.

The performance gains of the optimal scheduler have been pointed out by means of comparison with an empirical scheduler from the literature. Finally, the applicability of the proposed schemes has been discussed together with directions for the future development of the work.

References

1. Dinesh C. Verma, Hui Zhang, and Domenico Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proceedings of Tricomm '91*, Chapel Hill, North Carolina, Apr. 1991, IEEE.
2. Domenico Ferrari, "Distributed delay jitter control in packet-switching internetworks," Technical Report TR-91-056, University of California, Berkeley, California, Oct. 1991, also in *Journal of Internetworking: Practice and Experience.*
3. Reza Rejaie, Mark Handley, and Deborah Estrin, "An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.
4. Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Stockholm, Sweden, Aug. 2000, ACM.
5. James D. Salehi, Zhi-Li Zhang, Jim Kurose, and Don Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 397–410, Aug. 1998.
6. Z. Jiang and L. Kleinrock, "A general optimal video smoothing algorithm," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, California, March/April 1998, p. 676.
7. Zhi-Li Zhang, Yuewei Wang, D. H. C. Du, and Dongli Su, "Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, Aug. 2000.
8. Mario Baldi and Yoram Ofek, "End-to-end delay analysis of videoconferencing over packet switched networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, Aug. 2000.
9. William E. Naylor and Leonard Kleinrock, "Stream traffic communication in packet switched networks: destination buffering constraints," *IEEE Transactions on Communications*, vol. COM-30, no. 12, pp. 2527–2534, Dec. 1982.
10. Giulio Barberis and Daniele Pazzaglia, "Analysis and optimal design of a packet-voice receiver," *IEEE Transactions on Communications*, vol. COM-28, no. 2, pp. 217–227, Feb. 1980.
11. Warren A. Montgomery, "Techniques for packet voice synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, pp. 1022–1028, Dec. 1983.

12. Felipe Alvarez-Cuevas, Miquel Bertran, Francesc Oller, and Josep M. Selga, "Voice synchronization in packet switching networks," *IEEE Network*, vol. 7, no. 5, pp. 20–25, Sept. 1993.
13. Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Toronto, Canada, June 1994, pp. 680–688, IEEE Computer Society Press, Los Alamitos, California.
14. Sue B. Moon, Jim Kurose, and Don Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *ACM/Springer Multimedia Systems*, vol. 5, no. 1, pp. 17–28, Jan. 1998.
15. Maria C. Yuang, Shih T. Liang, Yu G. Chen, and Chi L. Shen, "Dynamic video playout smoothing method for multimedia applications," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*, Dallas, Texas, June 1996, p. S44.
16. Nikolaos Laoutaris and Ioannis Stavrakakis, "Adaptive playout strategies for packet video receivers with finite buffer capacity," in *Proceedings of the IEEE International Conference on Communications (IEEE ICC)*, Helsinki, Finland, June 2001.
17. Maria C. Yuang, Po L. Tien, and Shih T. Liang, "Intelligent video smoother for multimedia communications," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 136–146, Feb. 1997.
18. Hang Liu and Magda El Zarki, "Delay and synchronization control middleware to support real-time multimedia services over wireless PCS networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 9, pp. 1660–1671, Sept. 1999.
19. Mark Claypool and Jonathan Tanner, "The effects of jitter on the perceptual quality of video," in *ACM Multimedia '99*, Orlando, FL, USA, 1999.
20. Leonard Kleinrock, *Queueing Systems Volume I: Theory*, Wiley-Interscience, New York, 1975.
21. Ronald W. Wolff, *Stochastic Modeling and the Theory of Queues*, Prentice Hall, New Jersey, 1988.
22. Frederick S. Hillier and Gerald J. Lieberman, *Introduction to Operations Research*, McGraw-Hill, 2000.
23. O. Berman and K.P. Sapna, "Optimal control of service for facilities holding inventory," *Computers & Operations Research*, vol. 28, no. 5, Apr. 2001.

Constrained TCP-Friendly Congestion Control for Multimedia Communication

Dorgham Sisalem[1] and A. Wolisz[2]

[1] GMD FOKUS,Berlin, Germany,
`sisalem@fokus.gmd.de`
[2] TU Berlin/GMD Fokus, Berlin, Germany,
`wolisz@ee.tu-berlin.de`

Abstract. An often discussed approach for reducing overload situations in the network,is to adjust the transmission rate of the sender in accordance with the loss and delay state of the network. However, especially for the case of audio and video communication it is often the case, that the used compression style, application or the user might impose strict limitations as to the maximum and minimum transmission rates, the granularity with which the sender can change its transmission rate and the frequency with which such changes should occur.
In this paper, we present a generic model for describing a constrained multimedia source. Additionally, we describe a control framework called constrained TCP-friendly adaptation framework (CTFAF). CTFAF incorporates the proposed generic multimedia model with TCP-friendly adaptation schemes and hence allows for adapting the transmission rate of multimedia senders in a TCP-friendly way and at the same time takes the restrictions imposed by the source into account. Using simulations we investigate the performance of this approach in terms of bandwidth utilization, stability and fairness.

1 Introduction and Motivation

The rapid growth, increasing bandwidth and the availability of low-cost multimedia end systems have made it possible to use the Internet for multimedia applications ranging from telephony to conferencing, distance learning, media-on demand and broadcast applications [1].

However, transmitting such data packets into the network without regard to the actual available resources can easily result in overload situations and data losses. To avoid such a situation some mechanisms are required to control the amount of data packets entering the network.

An often discussed approach is to adapt the amount of data entering the network in accordance with the network congestion situation. This has been widely used in the Internet through the transport control protocol (TCP) used for data applications such as WWW and FTP transports. With TCP, the sender increases its transmission rate by an additive amount while no losses are observed. After observing a loss the rate is reduced by a multiplicative factor. To avoid overloading the network, non-TCP flows, e.g., multimedia flows, should deploy similar congestion control mechanisms to TCP. In addition to avoiding congestion and reducing the possibility of losses adaptation schemes for multimedia flows need to be fair towards competing TCP traffic, i.e., be TCP-friendly [2]. TCP-friendliness indicates here, that if a TCP connection and an adaptive flow with similar transmission behaviors have similar round trip delays and losses both connections should receive similar bandwidth shares. However, TCP shows a rather oscillatory adaptation behavior with fast and large changes in the transmission rate. Such a behavior results in an oscillative perceived QoS that can be rather annoying to the viewer of a video flow for example. Hence, multimedia flows require stable bandwidth shares that do not change on the scale of a round trip time as is the case of TCP connections. It is, thus, expected that a TCP-friendly multimedia flow would acquire the same bandwidth share as a TCP connection only averaged over time intervals of several seconds or even only over the entire life time of the flow and not at every time point [2].

Recently, there have been various proposals for realizing TCP-friendly adaptation [3,4,2,5].

All of those approaches assume that the sender can adjust its transmission rate in accordance with the values determined by the adaptation scheme and in accordance with the dynamics of the TCP-friendly algorithm with arbitrary granularity and with no boundaries as to the maximum or minimum bandwidth values. However, in reality there are certain maximum and minimum bandwidth requirements for multimedia contents above and below which the sent data is meaningless for the receiver. Additionally, the used compression style for a multimedia content might dictate that the changes in the transmission rate of a video stream for example can only be realized in a granular way. Also, there might be some restrictions arising from the user himself with regard to the acceptable variations in the perceived QoS over a time interval. Such restrictions depend primarily on the transfered content, the used compression style, the communication scenario and the user himself.

To investigate the effects of such restrictions on the adaptation process we present in Sec. 2 a simple model of a generic multimedia source and describe some constraints such a model might impose on the adaptation process. While this model does not represent a specific audio or video compression style, a communication scenario or particular user preferences, the model can be tuned through different parameters to describe a wide range of possible characterizations of multimedia sources and user preferences. Based on this model we describe in Sec. 3 a general framework called constrained TCP-friendly adaptation framework (CTFAF) to assist rate adaptation algorithms in incorporating various constraints in the adaptation process on the one hand and retain fair bandwidth distribution characteristics on the other. As an example of a TCP-friendly rate adaptation scheme we summarize in Sec. 4 the loss-delay based adaptation algorithm presented earlier in [5] called LDA+. With this approach, the adaptation scheme determines the actions of increasing or decreasing the transmission rate. The effects of enhancing an adaptation algorithm such as LDA+ to accommodate user and media constraints is then investigated in terms of bandwidth utilization, stability and TCP-friendliness in Sec. 5.

2 A Generic Model for Multimedia Communication

In designing an adaptation algorithm, we need to take the constraints imposed by the user and transmitted content into account. In this section, we present a generic model that describes possible constraints imposed by the user, compression algorithm and data content on an adaptation scheme. This model considers the following constraints:

Fixed limits: The quality of a multimedia stream can usually be improved by increasing the bandwidth share of the stream. However, above a certain limit ($r_{\max}$) no noticeable improvements in the QoS will be observed anymore. Additionally, reducing the bandwidth share of the stream below some minimal rate ($r_{\min}$) would redeem the transported data useless by the receiver. The values of those maximum and minimum values depend highly on the used compression styles, the users and the data content. While transmitting a talking heads kind of video conference at a frame rate of only a few frames per seconds might be acceptable, transmitting a live sports event requires obviously a high frame rate in order to be able to keep track of all instantaneous and rapid changes in the play field.

Granular adaptation: Depending on the used compression scheme and data content, a data source might only be able to change its transmission rate in steps of $S_{\text{adaptation}}$.

Taking the example of an MPEG video stream, changing the transmission rate can be achieved by either changing the transmission frame rate or the quantization level. In this case, the adaptation can only be realized in steps of ($m \times S_{\text{adaptation}}$), i.e., at the granularity of the discrete quantization steps or the size of a video frame. For the case of audio communication, adaptation can often only be realized by changing the used compression style which yields fixed adaptation steps as well.

Stable presentation: With TCP the transmission rate is changed in the order of once every round trip time in possibly large steps. Applying such an adaptation behavior to multimedia flows would result in a highly fluctuating perceived quality that can be annoying to the user. This can be observed for example when rapidly changing the frame rate for the case of video communication or continuously changing the used audio compression scheme.

To accommodate the requirement of a stable perceived QoS the transmission rate of a sender should not be increased or decreased by more than ($\delta_{\text{adaptation}}$: $\delta_{\text{adaptation}} \geq n \times S_{\text{adaptation}}$) during a time interval ($T_{\text{adaptation}}$), whereas $T_{\text{adaptation}}$ can be considered as the time between two adaptation points, i.e., the time points at which the sender executes the adaptation algorithm and adapts its transmission rate.

Loss tolerance: Depending on the transferred content, the user and the used compression scheme, some data losses might be tolerated during a multimedia communication. For example some compressions scheme already include loss recovery mechanisms for hiding data losses form the user. As another example, while users listening to an audio stream carrying music would require a very low loss rate, users involved in an interactive audio conference would be more tolerant to losses. Again here, the tolerated loss depends not only on the loss rate and distribution pattern but also on the fluency of the users in the used communication language for example. In such a case, the sender does not need to adjust its transmission rate if the loss value (l) was below a certain threshold (l_{allowed}).

Note, that this model only presents a subset of all the possible constraints that need to be considered when designing an adaptation scheme. Another example for such a constraint results from the case that changing some compression parameters must be followed by a recovery period to allow the senders and receivers to synchronize the encoding and decoding process in accordance with the new parameters.

3 A General Rate Adaptation Framework for Constrained Multimedia Flows

In this section, we present a general framework for adapting the transmission rate of constrained multimedia sources. With this framework, called the constrained TCP-friendly adaptation framework (CTFAF), the source uses a TCP-friendly adaptation algorithm for determining the bandwidth share of the multimedia flow. However, instead of adjusting the transmission rate of the source solely based on the rate determined by the control algorithm, the constraints imposed by the generic multimedia source as were described in Sec. 2 are taken into account as well.

Considering all the possible constraints imposed by multimedia compression styles and contents as well as the users in the adaptation process requires deep knowledge of the compression style as well as the content to be transmitted. Additionally, the minimum and maximum acceptable transmission rates result from the content and the communication scenario. Finally, the acceptable rate of changing the QoS depends also on the content as well as the user himself.

As a simplification of this problem we consider in this part of the work a generic encoder with the following characteristics:

– A minimum bandwidth requirement of r_{min} and a maximum one of r_{max}.
– The adaptation itself can only be realized in steps of ($m \times S_{\mathrm{adaptation}}$).
– To accommodate the requirement of a stable perceived QoS the changing rate
($\delta_{\mathrm{adaptation}} = \frac{\Delta r}{\Delta t}$) of the transmission rate of a sender should not be increased or decreased by more than ($n \times S_{\mathrm{adaptation}} \leq \delta_{\mathrm{adaptation}}$) during a time interval ($T_{\mathrm{adaptation}}$), whereas $T_{\mathrm{adaptation}}$ can be considered as the time between two adaptation points, i.e., the time points at which the sender executes the adaptation algorithm and adapts its transmission rate.
– A sender does not need to adjust its transmission rate if losses were below l_{allowed} or its transmission rate would go beneath r_{min} or above r_{max}.

As a generic adaptation algorithm we consider here a scheme that reduces the transmission rate of the sender by an amount X_{r} during congestion periods and increases it by an amount of X_{i} during network underload situations.

Now imposing constraints on the adaptation process means that during some time intervals the transmission rate of some flow can not be reduced even though losses are measured or the rate can not be increased even though the network is underloaded. To be able to integrate these constraints with the adaptation algorithm and still allow for a fair bandwidth distribution we incorporate the concept of virtual bandwidth share (B_{virtual}) with the adaptation process. A positive B_{virtual} represents the amount of bandwidth the flow should have acquired based on the calculations of some adaptation scheme ($r_{\text{calculated}}$) but did not use due to the constraints of the multimedia source. As the unused resources were consumed by competing flows an average equal bandwidth distribution can then be achieved if the flow consumed a bandwidth share larger than the rate calculated by the adaptation algorithm ($r_{\text{calculated}}$) in the future. A negative value indicates the amount of bandwidth used in excess of the amount calculated by the adaptation scheme and hence that the flow was too aggressive in the past and should use a transmission rate lower than $r_{\text{calculated}}$ when possible. While this approach does not guarantee fair bandwidth distribution at all times it allows for a fair distribution over long time periods.

With such an enhancement to an adaptation algorithm, the increase and decrease behavior of the control algorithm is influenced by the value of B_{virtual} in addition to the limitations imposed by the user or the data source and the network congestion state.

B_{virtual} is initialized to 0 at the beginning of the transmission process and then adjusted as follows:

– If the sender is allowed to increase its transmission rate by an amount larger than the maximum allowed value $\{X_{\text{i}} : X_{\text{i}} > \delta_{\text{adaptation}}\}$ then the sender increases its transmission rate by $\delta_{\text{adaptation}}$ and B_{virtual} is set to

$$B_{\text{virtual}} = B_{\text{virtual}} + X_{\text{i}} - \delta_{\text{adaptation}} \tag{1}$$

– If the sender needs to reduce its transmission rate by $\{X_{\text{r}} : X_{\text{r}} > \delta_{\text{adaptation}}\}$ then he only reduces its transmission rate by $\delta_{\text{adaptation}}$ and B_{virtual} is set to

$$B_{\text{virtual}} = B_{\text{virtual}} - X_{\text{r}} + \delta_{\text{adaptation}} \tag{2}$$

– If the sender needs to reduce its transmission rate by an amount X_{r} due to a loss (l) lower than the allowed loss value the flow can ignore or repair by himself $\{l : l < l_{\text{allowed}}\}$ or the reduction would lead to a transmission rate $\{r : r < r_{\text{min}}\}$ then he does not reduce its transmission rate and B_{virtual} is set to

$$B_{\text{virtual}} = B_{\text{virtual}} - X_{\text{r}} \tag{3}$$

An optimal adaptation behavior is realized with $B_{\text{virtual}} \approx 0$. To achieve such a behavior, in CTFAF, we additionally adjust the decrease and increase behavior of the adaptation algorithm to consider B_{virtual} and hence reduce the effects of temporarily unfair distribution as fast as possible.

Overload situation: For the case of network congestion the sender should reduce its current transmission rate by X_{r}.

Depending on the value of B_{virtual} the sender behavior is adjusted as follows:

$B_{\mathbf{virtual}} > 0$**:** A positive B_{virtual} indicates that this sender was too conservative in the past. As a compensation the sender can ignore the loss situation and maintain its current transmission rate. B_{virtual} is then reduced by the amount the rate should have been reduced by:

$$B_{\mathrm{virtual}} = B_{\mathrm{virtual}} - X_{\mathrm{r}} \tag{4}$$

$B_{\mathbf{virtual}} < 0$**:** A negative virtual bandwidth indicates on the contrary that the sender has been too aggressive in the past. Therefore, for the case of $X_{\mathrm{r}} < \delta_{\mathrm{adaptation}}$ the rate is reduced by $\delta_{\mathrm{adaptation}}$ and B_{virtual} can be increased as follows:

$$B_{\mathrm{virtual}} = B_{\mathrm{virtual}} + \delta_{\mathrm{adaptation}} - X_{\mathrm{r}} \tag{5}$$

Underload situation: For the case of network underload the sender would be allowed to increase its transmission rate by X_{i}. To compensate a negative virtual bandwidth the sender maintains in this case its current transmission rate. Thereby B_{virtual} is changed as follows:

$$B_{\mathrm{virtual}} = B_{\mathrm{virtual}} - X_{\mathrm{i}} \tag{6}$$

Recent values of B_{virtual} bear more significance in terms of characterizing the friendliness or aggressiveness of a flow towards competing traffic. Hence, to avoid having some value of B_{virtual} propagating over long time periods and influencing the behavior of the adaptation scheme even though the network situation that has resulted in this value has changed considerably we refer to resetting B_{virtual} in periods of T_{reset}. T_{reset} is several times as large as the adaptation intervals ($T_{\mathrm{adaptation}}$), i.e., the time between two adaptation actions. At the end of each resetting period (T_{reset}) the sender changes its transmission rate by an averaged B_{virtual} ($\overline{B_{\mathrm{virtual}}}$) and resets B_{virtual} down to zero. The averaged B_{virtual} is determined as

$$\overline{B_{\mathrm{virtual}}} = B_{\mathrm{virtual}} \times \frac{T_{\mathrm{adaptation}}}{T_{\mathrm{reset}}} \tag{7}$$

As B_{virtual} is changed at each adaptation point, $\overline{B_{\mathrm{virtual}}}$ represents an averaged value of all additions and subtractions done during T_{reset}.

Using a ($\overline{B_{\mathrm{virtual}}} : \overline{B_{\mathrm{virtual}}} > \delta_{\mathrm{adaptation}}$) contradicts the requirement of limiting the maximum changes in the transmission rate to $\delta_{\mathrm{adaptation}}$. However, we found this approach to be necessary in order to adjust the performance of a flow in accordance with the available resources. Without this resetting behavior, a sender transmitting data at a rate higher than the capacity of some link on the path towards the receiver and specifying a small $\delta_{\mathrm{adaptation}}$ compared to the needed change in the transmission rate would cause high and long lasting congestion. For the case that the flow is using a bandwidth share close to the average share calculated with the adaptation scheme, $\overline{B_{\text{virtual}}}$ would have a small value and would not have considerable effects on the adaptation behavior. Instead of using a large change of $(\overline{B_{\text{virtual}}} > \delta_{\text{adaptation}})$ another approach would be for the sender to change its transmission rate by $\delta_{\text{adaptation}}$ each $T_{\text{adaptation}}$ until B_{virtual} reaches 0 again.

Without prior knowledge of the available resources a sender might start transmitting data at a rate much higher than its fair share. If the sender additionally specified a slow maximum changing rate $\delta_{\text{adaptation}}$ the sent flow would lead to an overload situation lasting at least till the end of the first resetting period (T_{reset}). As T_{reset} might be in the range of one or more minutes relying solely on the resetting actions would take too long. To overcome this situation we specify that for a transient phase lasting for a maximum period of T_{init} the sender applies the adaptation scheme restricted only by the encoders limitations and not the user specifications. That is, for the first T_{init} seconds the sender can adapt its transmission rate by any number of adaptation steps $(S_{\text{adaptation}})$ as dictated by the adaptation scheme without regard to $\delta_{\text{adaptation}}$.

Note that the proposed guidelines here are only one option for accommodating user and media constraints into the adaptation process. For example, resetting B_{virtual} periodically is only one possible approach for preventing B_{virtual} accumulated during past periods of times to have long lasting effects on the adaptation process. Other options might be to use weighted moving averages or requiring that flows having a negative B_{virtual} value higher than some maximum value must reduce their rate more often and by larger amounts than indicated by the adaptation algorithm and user preferences. The exact strategies to use as well as their parameters will depend on the communication scenario.

4 The Enhanced Loss-Delay Based Adaptation Algorithm

As an example for a TCP-friendly adaptation scheme based on the increase/decrease concept, we describe here an adaptation algorithm that is based on previous work in [5] called the enhanced loss-delay based adaptation scheme (LDA+).

Most of the adaptation schemes presented in the literature [3,4] use feedback messages from the receiver sent in short intervals in the range of one or a few round trip delays and hence require a special control protocol.

On the contrary, LDA+ was designed to use the real time transport protocol (RTP) [6] for exchanging feedback information about the round trip time and the losses at the receiver. RTP defines a data and a control part. For the data part RTP specifies an additional header to be added to the data stream to identify the sender and type of data. With the control part called RTCP, each member of a communication session periodically sends control reports to all other members containing information about sent and received data. However, with RTP, the interval between sending two RTCP messages is usually around five seconds. The in-frequency of the RTCP feedback messages dictates that an RTP sender can not benefit fast enough from rapid changes in the network conditions. Thus, the goal of RTCP-based adaptation is to adjust the sender's transmission rate to the average available bandwidth and not react to rapid changes in buffer lengths of the routers for example. This might be actually more appropriate in some cases than rapidly changing the transmission rate at a high frequency. In addition to the inherent ability of RTCP to collect and distribute loss and delay information we extended it with the ability to estimate the bottleneck bandwidth of a connection based on the packet pair approach [7]. With this enhancement, the sender periodically transmits a number of data packets in bursts. Based on the time gaps (G) between two packets of size S the receiver can estimate the bottleneck bandwidth (B) as follows:

$$B = \frac{S}{G} \tag{8}$$

The information about which packets constitute the probing burst are included in the sender RTCP messages whereas the estimation results are included in an extension of the receiver reports.

Performance results obtained using simulations, comparisons to other schemes as well as measurements over the Internet suggest the efficiency of LDA+ in terms of TCP-friendliness and bandwidth utilization, see [5].

4.1 Rate Adjustment with LDA+

After receiving an RTCP message from the receiver indicating a no loss situation the sender increases its transmission rate (r) by an additive increase rate (R_{ai}). To allow competing flows of different bandwidth shares to faster converge to similar shares, R_{ai} is determined in dependence of the bandwidth share (r) the stream has already acquired relative to the maximally available share (R). Thus with an initial transmission rate of (r_0), an initial additive increase value of $\dot{R}_{\text{ai}}$, R_{ai} would evolve as follows:

$$R_{\text{ai}_1} = (1 - \frac{r_0}{R}) \times \dot{R}_{\text{ai}} \tag{9}$$

$$R_{\text{ai}_n} = (1 - \frac{r_{n-1}}{R}) \times R_{\text{ai}_n - 1} \tag{10}$$

Additionally, an RTP sender should not send more packets in between the reception of two RTCP messages than a TCP connection sharing the same link and having a similar round trip time would. So for an average period of T seconds in between the reception of RTCP messages, an initial rate (r_0) and a round trip delay of (τ) the TCP connection would send P packets with P set to

$$P = \sum_{i=0}^{T/\tau} w_0 + i \tag{11}$$

with the initial window ($w_0 = r_0 \times \tau$) and the window size is increased by one packet each round trip delay. Thus the RTCP sender should maximally increase its transmission rate to

$$r = \frac{P}{T} \rightarrow \frac{(\frac{T}{\tau} + 1)}{2} \tag{12}$$

With RTCP the round trip delay (τ) is estimated by including a timestamp in the sender reports. The receivers include in their reports the timestamp of the last seen sender (t_{DLSR}) report and the time passed between receiving that report and sending the receiver report (t_{LSR}). Thus τ can be estimated at the sender as the difference between the current time and ($t_{\text{DLSR}} + t_{\text{LSR}}$).

For the case of losses, the work done in [8] suggest that TCP adjusts its transmission rate inversely proportional to the square root of the losses. Thus, if losses (l) were indicated in the RTCP messages then the transmission rate is reduced to:

$$r = \min(r \times (1 - \sqrt{l}), r_{\text{TCP}}) \tag{13}$$

with r_{TCP} as the rate calculated using the TCP model presented in [8]. Additionally, R_{ai} is set to $\dot{R}_{\text{ai}}$. In case the current transmission rate is lower than r_{TCP} the sender is allowed to further increase its transmission rate up to r_{TCP}.

5 Performance Investigation of CTFAF

In this part of the work, we investigate the effects of imposing user- or encoder-based restrictions on the adaptation process in terms of buffer occupancy and bandwidth utilization. For this purpose we compare the performance of LDA+ without any restrictions, i.e., ($\delta_{\text{adaptation}} = \infty$) and its performance when the rate adjustments are controlled not only by network conditions but by the CTFAF guidelines described in Sec. 3 as well.

For investigating the performance of CTFAF we used the simulation topology depicted in Fig. 1 with FTP, WWW and multimedia flows competing for a bottleneck router of 10 Mb/s and ($m = n = k = 27$). The used topology and parameters resemble to a great extent the topologies used in a large number of papers investigating the issue of TCP-friendly congestion control [4,3]. The FTP and multimedia flows were modeled as greedy sources that can always send data at the rate allowed by the congestion control scheme, i.e., the TCP mechanisms for the case of FTP and CTFAF for the case of multimedia flows. The WWW servers were modeled as on-off processes with the on period lasting for the time needed to transmit a number of packets drawn from a Pareto distribution with the factor of 1.1 and a mean of 20 packets and the off period lasting

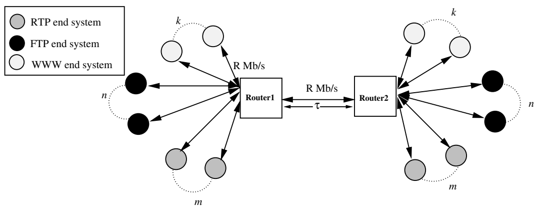


Fig. 1. CTFAF performance testing topology

for a time drawn from a Pareto distribution with a factor of 1.8 and a mean of 0.5 seconds [9].

The routers used random early packet discard (RED) [10] buffer management. A RED gateway detects incipient congestion by computing the average queue size. When the average queue size exceeds a preset minimum threshold the router drops each incoming packet with some probability. Exceeding a second maximum threshold leads to dropping all arriving packets. This approach not only keeps the average queue length low but ensures that all connections receive the same loss ratio and avoids synchronization among the competing flows. This is of utmost importance in this study in order to properly evaluate the fairness aspects of the investigated adaptation schemes on connections having the same loss. However, the performance of the here presented solutions does not depend on the usage of RED routers. Based on results achieved in [11] the minimum drop threshold was set to 0.3 of the router buffer and the maximum to 0.8.

Unless otherwise stated, the round trip propagation delay was set to 0.4 seconds. The initialization period (T_{init}) was set to 60 seconds. The performance of CTFAF was tested in terms of achieved average rate (r), fairness factor (F), average standard deviation (σ) and average utilization value (u). The average rate (r) was determined over the entire simulation time minus the transient time which was approximated in this case to the first 200 seconds. The standard deviation of the transmission rate of the single flows was used as an indication of the stability of the adaptation scheme. Similar to [3], the TCP-friendliness (F) of CTFAF was determined as

$$F = \frac{r_{\text{CTFAF}}}{r_{\text{TCP}}} \tag{14}$$

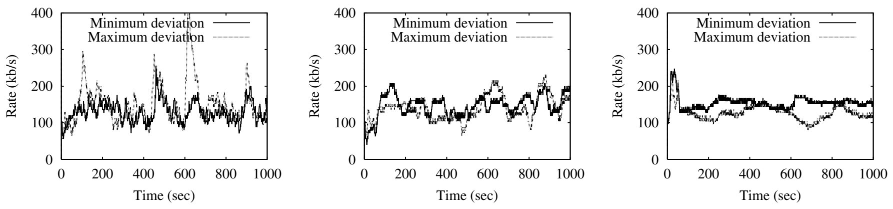
with r_{CTFAF} as the average goodput of the RTP flows using the CTFAF framework and r_{TCP} as the average goodput of the TCP connections.

5.1 Effects of the Length of the Adaptation Steps ($S_{\text{adaptation}}$) on the Performance of CTFAF

As an example for a multimedia flow we simulated the case of video flows with constant quantization steps of 2 kbits, packet sizes of 10 kbits and frame sizes of 20 kbits. We tested the case of coarse and fine granularity adaptation. Using ($S_{\text{adaptation}}$ = 20 kb/s) resembles the case of changing the transmission rate by varying the number of sent video frames when using JPEG compression for example. We set $\delta_{\text{adaptation}}$ in this case to 1 and 2 frames/s, i.e., 20 and 40 kb/s. With ($S_{\text{adaptation}}$ = 2 kb/s) the adaptation is more granular and simulates the case of varying the quantization or quality factor. $\delta_{\text{adaptation}}$ is set in this case to 4 kb/s. As a minimum transmission rate we use a value of 40 kb/s. The parameter settings for the video coder were chosen based on observations made during a talking heads conference with JPEG as the compression style. For another communication scenario or a different compression style we would most likely have to use a different set of parameters. Refining the video model and choosing the appropriate parameters requires detailed knowledge of the used compression

Table 1. Achieved average rate and fairness of CTFAF with different values of $\delta_{\text{adaptation}}$

$\delta_{\text{adaptation}}$ (kb/s)	r (kb/s)	σ (kb/s)	F	u
∞	134	44	0.8	0.81
40	137	45	0.84	0.80
20	132	37	0.80	0.79
4	140	29	0.85	0.81



(a) $\delta_{\text{adaptation}} = \infty\,\text{kb/s}$ (b) $\delta_{\text{adaptation}}{=}20\,\text{kb/s}$ (c) $\delta_{\text{adaptation}}{=}4\,\text{kb/s}$

Fig. 2. Temporal behavior of CTFAF flows with the maximum and minimum standard deviation values for different values of $\delta_{\text{adaptation}}$

style and communication scenario. Additionally, to account for the user preferences extensive subjective testing is needed. The resetting period (T_{reset}) was set to 60 seconds.

From Tab. 1 we can observe that while the standard deviation which indicates the level of oscillations of the flows is reduced for finer setting of $\delta_{\text{adaptation}}$ the utilization and fairness levels are hardly changed compared to the case of no restrictions on the adaptation ($\delta_{\text{adaptation}} = \infty$).

The reduction of oscillations is particularly evident in Fig. 2 which show the temporal behavior of the flows with the largest and smallest standard deviation (σ) values. For the case of ($\delta_{\text{adaptation}} = 4\text{kb/s}$) the flows have a much smoother shape compared to the other cases.

The fairness results presented in this section suggest that while CTFAF maintains the TCP-friendliness of LDA+, it can successfully take various constraints imposed by the multimedia flow into account.

5.2 Effects of the Length of the Resetting Period (T_{reset}) on the Performance of CTFAF

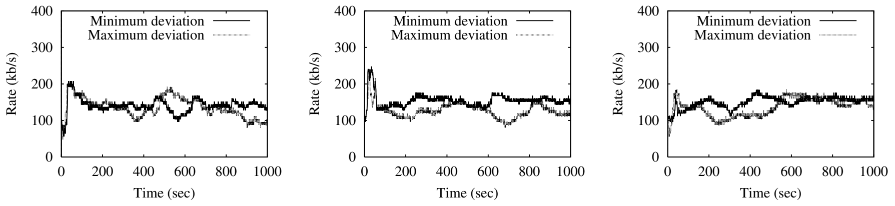
The length of the resetting period (T_{reset}) determines the interval in which B_{virtual} is reset and the bandwidth share of the flow is corrected possibly faster than the user specified adaptation rate ($\delta_{\text{adaptation}}$). Setting T_{reset} too small would mean that this correction is done rather often and would, thus, defy our goal of restricting the adaptation rate to $\delta_{\text{adaptation}}$. Using too large values of T_{reset} would on the other hand mean that the flow can not adapt rapidly enough to large changes in the network conditions.

Here, we simulated the case of competing WWW, FTP and CTFAF flows over a link of 10 Mb/s with a round trip propagation delay of 0.4 seconds and a maximum queuing delay of 0.1 seconds. The CTFAF flows used adaptation steps ($S_{\text{adaptation}}$) of 2 kb/s and the maximum possible rate change at each adaptation point ($\delta_{\text{adaptation}}$) was set to 4 kb/s. These adaptation values impose rather strict constraints on the adaptation process and a better distinction to the case of adapting the transmission rate without any restrictions ($\delta_{\text{adaptation}} = \infty$).

Table 2. Achieved average rate and fairness of CTFAF with different values of T_{reset}

T_{reset} (sec)	r (kb/s)	σ (kb/s)	F	u
30	137	30	0.81	0.82
60	140	29	0.85	0.81
120	134	26	0.83	0.81
300	142	25	0.86	0.83

The fairness and utilization results in Tab. 2 show little variance with varying values of T_{reset}. The average standard deviation values (σ) presented in Tab. 2 as well as the temporal behavior of the flows with the best and worst deviation values in Fig. 3 suggest that with increased values of T_{reset} the flows display a lower degree of oscillations. However, with large values of T_{reset} the flows can not react to large and sudden changes in the network conditions such as when several new flows start sending data on the same link as the CTFAF flows. Hence, setting T_{reset} to 60 or 120 seconds presents a better compromise between stability and fast reaction than a value of 300 seconds for example.



(a) $T_{\text{reset}} = 30$ sec (b) $T_{\text{reset}} = 60$ sec (c) $T_{\text{reset}} = 120$ sec

Fig. 3. Temporal behavior of CTFAF flows with the maximum and minimum of σ for different values of T_{reset}

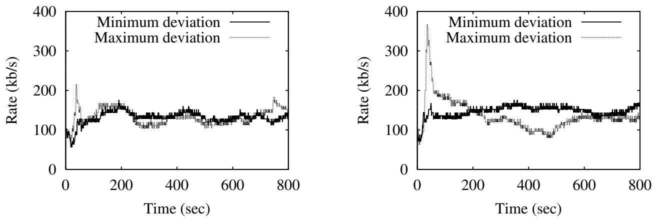
5.3 Effects of Delays and Flow Numbers on the Performance of CTFAF

In this part, we investigate the performance of CTFAF in terms of bandwidth utilization (u), average bandwidth share of single CTFAF flows (r) and its fairness towards competing TCP connection (F) under different test settings with varying numbers of competing flows and delays. As a simulation topology we used the one described in Fig. 1 with a varying number of flows competing for a bottleneck of 10 Mb/s. In addition to n FTP and m CTFAF flows, 27 WWW servers continuously generated short TCP connections.

Tab. 3 presents the simulation results for the case of varying number of competing flows ($m = n = N$) and round trip propagation delays (τ). From Tab. 3 we can observe that the average standard deviation of the flows (σ) is around than 20% of the average bandwidth of the flows. Additionally, the deviation ($\overline{\sigma}$) in the average rate values of the single flows from the average bandwidth share of all the flows is rather negligible indicating a high degree of inter-flow fairness with all the flows receiving similar bandwidth shares. The fairness index (F) is in an acceptable range between 0.8 and 1.15 indicating similar bandwidth shares for the TCP and CTFAF flows.

Table 3. Achieved average rate (r) in kb/s, deviation (σ,$\overline{\sigma}$) in kb/s, utilization (u) and fairness of CTFAF with N TCP and N CTFAF competing flows, R set to 10 Mb/s and τ_q set to 0.1 seconds

N	27					81				
τ	r	σ	$\overline{\sigma}$	F	u	r	σ	$\overline{\sigma}$	F	u
0.1	160	33	7.8	0.9	0.9	75	19	4	1.3	0.9
0.2	130	27	7.5	0.7	0.8	62	16.1	4.1	1	0.9
0.4	140	29	13	0.8	0.8	55	15	2.1	0.86	0.9
0.6	169	32	16	1.1	0.8	54	13.9	5	0.8	0.9



(a) $\tau = 0.2, \tau_q = 0.1$ (b) $\tau = 0.4, \tau_q = 0.1$

Fig. 4. Temporal behavior of CTFAF for N set to 27

6 Summary and Future Work

In this paper, we presented a novel approach for incorporating user and coder imposed restrictions on the adaptation behavior with a TCP-friendly adaptation algorithm used for controlling the transmission of multimedia flows. The achieved simulation results suggest that the used model and adaptation approach maintain the TCP-friendliness characteristics of the original algorithm and achieve a stable adaptation behavior that corresponds to the parameters imposed by the coder and the user.

While the work presented here gives a possible approach for integrating congestion control with multimedia communication, there are still many open questions regarding the exact tuning of the adaptation parameters, incorporating different kinds of compression styles and the actual implications of such an adaptation behavior on the perceived quality. This work should be seen as work in progress that requires simulating a wider range of possible parameters as well as assessing the changes in perceived quality of service on real life multimedia streams using series of subjective tests. Further, the case of congestion control in multicast communication [12] with heterogeneous receiver requirements and capabilities needs to be studied.

References

1. Henning Schulzrinne, "Re-engineering the telephone system," in *Proc. of IEEE Singapore International Conference on Networks (SICON)*, Singapore, Apr. 1997.
2. Mark Handely and Sally Floyd, "Strawman specification for TCP friendly reliable multicast congestion control," 1998, Note to the Internet Reliable Multicast Group mailing list.
3. Reza Rejaie, Mark Handley, and Deborah Estrin, "An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Infocom'99*, New York, March 1999, IEEE.
4. Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Stockholm, Sweden, Aug. 2000.
5. Dorgham Sisalem and Adam Wolisz, "LDA+ TCP-friendly adaptation: A measurement and comparison study," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Chapel Hill, NC, USA, 2000, pp. 183–192.
6. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Tech. Rep. RFC 1889, Internet Engineering Task Force, Jan. 1996.
7. Jean-Chrysostome Bolot, "End-to-end packet delay and loss behavior in the Internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Deepinder P. Sidhu, Ed., San Francisco, California, Sept. 1993, ACM, pp. 289–298, also in *Computer Communication Review* 23 (4), Oct. 1992.
8. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *ACM SIGCOMM '98*, Vancouver, Oct 1998.

9. Kihong Park, Gitae Kim, and Mark Corvella, "On the relationship between file sizes, transport protocols and self-simi lar network traffic," in *International Conference on Network Protocols (ICNP)*, Columbus, Ohio, Oct 1996.
10. Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
11. Sally Floyd and Fall Kevin, "Router mechanisms to support end-to-end congestion control," Tech. Rep., LBL-Berkeley, Feb. 1997.
12. Dorgham Sisalem and Adam Wolisz, "MLDA: A TCP-friendly congestion control framework for heterogeneous multicast e nvironments," in *Eighth International Workshop on Quality of Service (IWQoS 2000)*, Pittsburgh, PA, June 2000, IEEE, pp. 65–74.

Adaptive Wavelet Video Filtering

Andreas Kassler[1], Christian Kücherer[2], and Andreas Schrader[2]

[1] University of Ulm, Distributed Systems Department, Ulm, Germany
`Andreas.Kassler@informatik.uni-ulm.de`,
[2] NEC Europe Ltd., Network Laboratories, Heidelberg, Germany
{`Christian.Kuecherer` | `Andreas.Schrader`}`@ccrle.nec.de`

Abstract. In this paper we propose a flexible adaptive media streaming mechanism based on wavelet encoded video streams. By using a combination of sender rate adaptation and filtering inside the transmission path, an appropriate tradeoff between flexibility, efficiency and security can be achieved. The media adaptation mechanisms are one part of the MASA Quality-of-Service framework, which provides end-to-end QoS enhanced multimedia communication and includes local resource management. We provide examples for syntactial and semantical filter algorithms, show the interaction of media adaptation and QoS management and present experimental measurement results.

1 Introduction

The Internet offers challenging new opportunities for the development of communication applications such as real time video conferencing, multimedia distance learning and many more. The growing number of different fixed line (e.g. Ethernet, ATM, xDSL, etc.) and wireless (e.g. GSM/GPRS, Wireless LAN, UMTS, etc.) access network technologies in combination with the growing variety of user devices (e.g. mobile phone, PDA, Multimedia Workstation, etc.) will lead to a high level of heterogenity. Another source of heterogenity is imposed by the fact, that users should have the possibility of defining their own QoS policies with parameters like color resolution, frame size or frame rate.

In each of these categories, static and dynamic variations can occur. Transmission links do have a maximum available bandwidth, but the actually available bandwidth will vary over time depending on the overall usage. Especially, when mobile terminals perform intra- and inter-technology handoffs, transmission link characteristics will show sudden significant variations. Similar dynamic variations could also appear for terminal capabilities and user QoS policies.

Therefore, adaptivity is necessary to provide a high flexibility. Adaptation of media streams is the most important task as it influences user perceived quality as well as resource requirements both for the endsystems and for the network. Sender rate adaptation offers the highest flexibility, since the full semantical information of the data source is available. In addition, network node based media adaptation can adapt the stream to the aggregated demands of the following subtree, which bridges the heterogeneity gap for multicast scenarios. Mid-term, slow adaptation to the static demands of the session can be achieved by using appropriate feedback mechanisms in a distributed set of media adaptation nodes. To deal with highly dynamic changes during the transmission, simple filtering mechanisms inside routers should be used in addition to allow for short-term, very fast, locally optimized reactions and seamless handoff performance on mobile clients (e.g. in case of queue buffer overflows). Finally, user perceived QoS needs to be managed and maintained end-to-end. This requires local, peer and network resource reservation, adaptive media magement to react properly to QoS violations and a coordination of the activities to comply to users requirements. As it is too complex to manage these tasks by an application itself, a QoS framework (like the MASA QoS framework [5]) is necessary that applications can use to provide QoS enhanced adaptive multimedia services towards the user.

Several adaptive codecs have been proposed (e.g. layered DCT [2], *lH.261* [3]), but they often suffer from the flexibility for the support of different receiver requirements. Recently, Wavelet based coding mechanisms have become very popular [1]. In this paper we are using the WaveVideo codec developed at the ETH Zürich [4], which offers a good compromise between compression ratio, error tolerance, implementation efficiency, and the ability to support various filter operations. By producing a layered hierarchical coding scheme, WaveVideo is able to support prioritization of packets.

There are different possibilities to realize scaled media streaming. In the receiver driven layered multicast (RLM) approach, proposed in [11], receivers can decide about the quality of the stream by joining or leaving the respective multicast groups, which contain different layers of information. However, the most important problem of fixed layer subscription schemes is the unability to support different receiver QoS policies at the same time without inserting redundant layers. Therefore, we propose to use a single (multicast) stream which contains the aggregated sum of all requested data parts for the respective subtrees.

In this paper we propose two different video adaptation strategies. A simple, but efficient filter is based on pure syntactical information. A set of semantical filters allow for highest flexibility. We will demonstrate the use of both filter types in a multicast scenario with a single QoS policy and in a unicast scenario supporting multiple QoS policies.

This paper is organized as follows. In section 2, we give an overview of the MASA QoS framework. A set of syntactical and semantical filter algorithms for WaveVideo streams is presented in section 3. In section 4 we describe two example implementations of our adaptive media management system. Section 5 reports on some first measurement performed for two application scenarios in a wireless environment. Finally, we conclude our paper and present future work.

2 The MASA QoS Framework

The *MASA* QoS framework (Mobility and Service Adaptation in Heterogeneous Mobile Networks) is a joint project of NEC Europe Ltd. Heidelberg, Siemens AG Munich and the University of Ulm [5]. The *MASA* framework was designed to fulfill the requirements of a *comprehensive integrated end-to-end QoS multimedia management system*, allows the usage of underlying network layer QoS technologies, and hides the complexity of QoS and adaptive media management from the applications. By controlling the complete communication infrastructure *MASA* is able to support QoS in a way which can neither be realized inside the applications nor with the underlying network QoS technologies alone.

2.1 Architectural Overview

The *MASA* QoS architecture consists of a distributed set of autonomous QoS Brokers (fig. 1) which can be placed on the (potentially mobile) end-system, on intermediate network nodes (e.g. router, switches) and on transcoding units (gateways). Each Broker delegates and coordinates different Managers which in turn are responsible for specific tasks, like resource, network, media, monitoring, policy and mobility management.

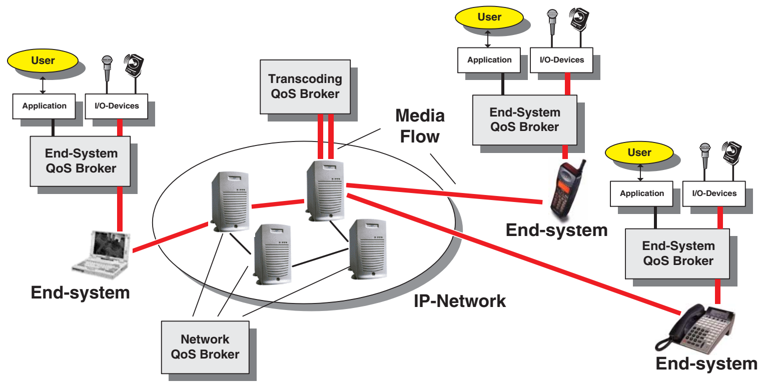


Fig. 1. Distributed set of autonomous QoS Brokers.

The main task of the *End-System QoS Broker* is to coordinate, orchestrate and manage local and remote resources for adaptive multimedia streaming, to map the user's QoS wishes to appropriate QoS parameters and to support mobility with the use of different access networks (e.g. GSM, WirelessLAN, UMTS, etc.). *Network QoS Brokers* can be regarded as centralised QoS management units which support the End-System QoS Brokers and organise the orchestration of all streams in the respective network domain according to a given network management policy. The Network QoS Broker monitors network resources, decides admission in cooperation with other admission controlling entities (e.g. H.323 Gatekeepers) and realises load balancing and fairness concepts for all participating systems. The actual behaviour of the Network QoS Broker depends on the location of the respective network node, e.g. core network router, access network router, media gateway, LAN switch etc. The *Transcoding QoS*

Broker coordinates its Managers to provide services for media adaptation which may include changing the codec to support heterogeneous clients. In addition, adaptive Forward Error Correction (FEC), appropriate transmission protocols (e.g. wireless-TCP), etc. may be invoked. Transcoding QoS Brokers allow for automatic downloading of codecs and media adaptation objects on demand. A detailed description of the network QoS Broker and the transcoding QoS Broker is outside the scope of this paper.

Communication between the distributed collection of QoS Brokers is realized via appropriate interfaces. Main communication issues are capability exchange methods, QoS routing mechanisms, admission and authorization requests, and the management of media channels.

MASA presents applications with mechanisms for the processing and transmission of 'high quality' multimedia streams, i.e. adapted to the user's QoS wishes and the available infrastructure and resource availability. Applications subscribe to the system and use the provided multimedia facilities to control the set-up of a complete chain of media processors, consisting of capture devices, codecs, effect processors, etc. Appropriate graphical user interfaces are provided to present media information (e.g. video panel).

Efficient handoff algorithms are a cost-effective way of enhancing the capacity and QoS of cellular systems. *MASA* supports mobile devices by integrating mobility management into the framework, using fast QoS re-negotiation and adaptation mechanisms to allow seamless intra- and inter-technology handoff.

2.2 Hierarchical QoS Broker Structure

Fig. 2 outlines the typical structure of a *MASA* QoS module. The *QoS Broker* is the central intelligence unit which is supported by a set of *QoS Managers* which in turn are supported by appropriate *QoS Controllers*. With this hierarchy of Manager/Controller structures, the QoS Broker can delegate separate tasks for controlling and processing media streams and therefore provides a clear separation of tasks with different time constraints.

The *Policy Manager* is responsible for the storage and retrieval of QoS preferences within a user profile. It presents an appropriate policy GUI to the user. The Policy Controller enables the access to a policy database. The *Resource Managers* are responsible for controlling the available resources (like CPU, memory, network, etc.) via the respective Resource Controllers. The *Media Manager* is responsible for the provision and orchestration of actual media processing entities, like codecs, packetizers, etc., inside the Media Controller. It also monitors the transmission parameters and reports aggregated statistical information to the Broker. The *Intercom Manager* is used to support inter-Broker communication. The *Application Manager* provides mapping functionality between different categories of applications, like VoD or IP-Telephony and the Broker QoS API. The *Mobility Management* is responsible for the support of device mobility and enables the usage of different access devices. We have implemented a Mobility Manager using Mobile IP [6]. The Mobility Controller is a standard Mobile IP daemon. For each network interface an Access Network Monitoring

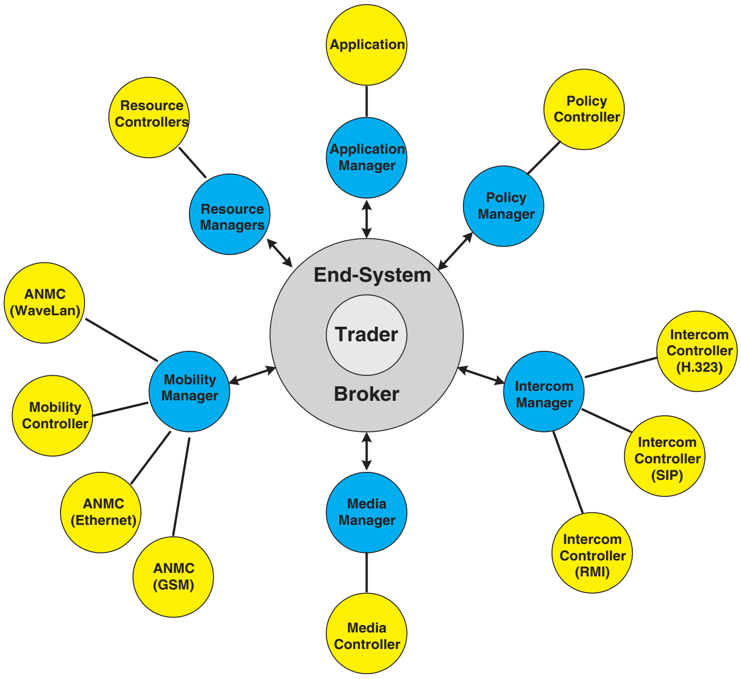


Fig. 2. Hierarchical structure of *MASA* on end-systems.

Controller (ANMC) measures link quality parameters and reports them to the Mobility Manager, which either directly forces a handoff or informs the QoS Broker about available network options. This tight integration of mobility management and handoff control within the QoS architecture allows for supporting seamless handoffs, fast adaptation (e.g. by changing filter settings, see section 4) as well as QoS-controlled handoff decisions.

The Broker regularly requests monitoring information from its Managers. The aggregated monitoring information together with the user's QoS policy is used as input for a trader mechanims which analysis the current situation and decides for possible adaptation of the current active sessions. On the end-system, the algorithm of the trader is controlled by a local trading policy which can be easily exchanged even during runtime. The Broker parses the result and informs the respective Managers about the necessary actions that have to be performed to realize the adaptation. Examples are codec changes inside the Media Manager or handoffs inside the Mobility Manager.

Since not all Managers have to be used for each Broker type, our design provides scalability. For example, at transcoding/filtering nodes, the Broker does not need Application or Mobility Managers. Through the use of open interfaces between the QoS Brokers and Managers, the system can be easily extended with new Manager/Controller pairs. For example, a Manager/Controller pair for power management would allow to support a trading policy like *when the* *battery power is low, switch to codec X.* The major parts of the architecture have been implemented in Java to allow downloadability of new plug-ins even during runtime. The interfaces between Broker and Managers have been defined by using two request/response queues for each pair, allowing for asynchronous non-blocking processing of requests with different time constraints. A standard interface consisting of methods like *adapt(), get_monitor_results(), etc.* can be extended for each type of Manager.

3 Adaptive Media Management

As basis for our adaptive media management we use an adaptive media coder called WaveVideo [4], which is based on the wavelet transform and offers a high degree of error tolerance as well as a high compression ratio at a good image quality. Media adaptation is achieved by inspecting a 32-bit-tag inserted by the coder which describes the content of each network packet. As an example, the tag allows the decoder or any media filter (see section 4) to associate each WaveVideo packet to its frame and for each frame to determine the correct colour/luminance plane. It also determines the associated leave (combination or recursion depth and direction) of the wavelet tree and the quantization layer to which the coefficients contained in the packets contributes to. Video adaptation is achieved by dropping network packets that do not contribute to a desired quality. For more information about the WaveVideo coding method see also [4].

The Adaptive Media Management is implemented by the Media Controller, which is configured by the Media Manager on requests from the Broker. The QoS Broker derives a QoS contract for a given stream based on user QoS parameters maintained by the Policy Manager/Controller. The mapping is given in [10]. It is up to the Media Controller to enforce the QoS contract. Typical parameters are frame rate, frame size, visual quality and end-to-end delay/jitter. The Media Controller at the receiver maintains playout buffers to control the delay, jitter and losses and may request re-transmission of lost packets from the sender Media Controller. If the QoS contract is violated, the receiver Media Controller informs the Broker, which requests media stream adaptation (see section 4) to react to resource availability.

3.1 Java Media Framework Plugin

The NEC *AQUARIUS* Media Manager is built in Java. The Media Controller is implemented as an extension of the Java Media Framework (JMF) [12]. JMF is an optional Java package, providing APIs for using, manipulating and managing audio, video and other time-based media. It can be expected that JMF will be supported in the near future by small handheld devices. Therefore, JMF is well suited for mobile multimedia communication.

Through the usage of JMF *AQUARIUS* is able to download, install and maintain a broad range of audio/video codecs to filter and transmit media (via RTP), ranging from low bandwidth speech codecs (e.g. GSM) to high quality video streaming (e.g. H.261/H.263, MPEG-1/2/4). Additional custom coding schemes can be integrated into the architecture using multiplexer, demultiplexer, codec, effect filters or renderer plug-ins.

To support WaveVideo RTP streaming, we have implemented packetizer and depacketizer JMF plug-ins. The packetizer splits WaveVideo encoded frames into network packets based on the tag information and sends them over the network by using the JMF RTP Manager. The depacketizer collects the packets, reconstructs the video frame and gives it to a codec and video renderer for presentation. We have also implemented a JMF filter plug-in, which allows to integrate filter algorithms into the media transmission chain of JMF. The filter plug-in can be parametrized by the supported quality interface, allowing to specify a quality value in $[0, 1]$ (see section 5).

3.2 C++ Implementation

Typically, distributed multimedia applications should be able to operate in real time. This was motivation for the University of Ulm to implement a Media Controller in native C++ which realizes high performance media processing and adaptation. Since the whole MASA framework is implemented in Java, interoperability has been achieved by implementing the Media Manager in Java and providing a simple wrapper to the functionality provided by the native Media Controller as well as a data converter. The Media Manager is therefore a stub, which interacts at the Java side with the Broker's requests, which are propagated to the native Media Controller using customized socket communication. The Media Controller itself provides mechanisms to open, close and (re-)configure a local or remote media channel. Monitoring data is provided on request and forwarded to the Broker via the Media Manager. In the case that the requested quality can no longer be maintained, the Media Manager informs the Broker, which starts a new trading cycle based on the monitoring information and users QoS wishes to decide, if new filter settings have to be applied or the codec needs to be changed.

4 Filtering Algorithms

In this section we will describe some example filter algorithms for WaveVideo codec video streams. Simple priority-based filtering based on syntactical information can be used to efficiently support a single adaptation policy in a multicast scenario. More complex structure-based filtering based on the semantical information of the WaveVideo tag can be used for a unicast scenario, allowing for different receiver adaptation policies.

All filter algorithms are implemented inside the Media Controller under the control of the MASA QoS framework. Access to the filter services is provided at the end-system using the End-System QoS Broker or at the transcoders using the Transcoding QoS Broker. Filter services can be instantiated on any video stream maintained by the Media Controller. (Re-)configuration of the filter algorithms is achieved by the QoS Broker via the Media Manager based on decisions of the trading policy or user input (e.g. user changes its requirements).

Note, that the filter algorithms do not depend on the MASA architecture. In principle, they work on any WaveVideo compressed stream. However, the algorithms were implemented under the MASA framework.

4.1 Syntactical Filtering

The term syntactical filtering is used for algorithms, which base their packet dropping decisions on simple structural information, e.g. priority fields. This scheme can only support one single adaptation policy, since the priority fields can not reflect the relative importance of the packet for the respective users.

One simple filter algorithm is the following. A given input frame consists of n network packets, each carrying one quantization layer of every subband of every color/luminance plane. The number of packets m to be sent is calculated with the quality function f_q defined as follows:

$$f_q : m = trunc(n * q)$$

whereas q is the quality factor in $[0, 1]$, n is the number of packets for this frame at a 100% quality and m the number of packets (starting at packet 0) to be sent with $m \leq n$. Thus, the first m packets of a frame ordered by priority will be passed and the least important packets ($packet_m \ldots packet_{n-1}$) will be discarded. The two extremes of this function are:

- $q = 1$:
  every packet in the input frame which can pass the filter is copied to the output frame.
- $q = 0$:
  none of the packets in the input frame can pass the filter. No output frame is created.

4.2 Semantical Filtering

In this section we classify semantical filter algorithms and show how they can be applied to WaveVideo streams. Transcoder filters have been presented and evaluated in [9]. In addition, a dynamic rate shaping filter (DRS-P) that includes user preferences has been presented in [7].

Filters adapting the frame rate operate in the temporal domain and reduce the frame rate. The frame number and the frame type (I- or Delta-frame) are coded within the WaveVideo tag. The frame rate generated by the source is included in an I-frame header [4]. Let us suppose that the sender sends at r_{src} and the receiver wants to be served with r_{dest}. The frame rate filter has to decide, if the frame k has to be dropped based on r_{src} and r_{dest} and the dropping history (i.e. what frames have already been discarded).

Filters adapting visual quality of single video frames operate in the frequency domain, degrade visual quality of the luminance and/or chrominance planes and reduce bandwidth without affecting frame rate. A filter adapting colour quality reduces the visual quality of the chrominance planes only. The WaveVideo frame quality filter [9] first extracts frequency information and then determines which coefficients to discard. Filtering is achieved by dropping whole subbands (or certain quantization layers).

The combined filter offers filter services to adapt frame rate, frame size, luminance quality and chrominance quality independently and is controlled by:

– resolution r_l (0 denotes best Quality), i.e. quality of the luminance channel
– quality of the chrominance channels r_c (10 denotes black/white)
– frame size r_s.

Variation of r_l and r_c influences the visible quality by implementing subband dropping. Subbands are not dropped if $r_l = r_c = 0$. All subbands i (i.e. all packets contributing to subband i) referring to the luminance plane are dropped with $i \leq r_l$. For the colour resolution similar rules apply [9]. The combi-filter allows to reduce frame width and height by a factor of two by setting $r_l = r_c = 2$. The new frame size is calculated to $width_{new} = \frac{width_{old}}{2^{r_s}}$ and $height_{new} = \frac{height_{old}}{2^{r_s}}$. Finally, an I-frame header is updated to notify the decoder about the new dimension of the frames. Frame-rate filter services are provided in the combi-filter by including a new parameter *target frame rate* r_{dest} $(0 < r_{dest} \leq r_{src})$. The combi-filter first performs frame rate filtering as described above. All remaining frames are then filtered according to the settings of r_l and r_c.

5 Measurement Results

In this section we present initial experimental results for the proposed filter algorithms in a real environment using the MASA QoS framework on a testbed consisting of mobile clients connected via various access network technologies, like e.g. WaveLan and Ethernet.

The performance of the semantical filters have already been evaluated in [9], [7] and [8]. In this work we present additional performance values for the syntactical filter and focus on the interaction between QoS Broker and filter algorithms implemented inside the Media Controller. In particular we are interested in adapting the media stream at the sender side based on various feedback information. To this extend, we focus on the *trading policy* inside the Broker and its interaction with the media management. The decision to reconfigure filters is made by the Trader based on measurement results, which have been obtained from all Managers, QoS profiles and preferences provided by the Policy Manager and resource availability.

We implemented two video application scenarios for unicast as well as multicast transmission using the Application Manager to request adaptive media streaming and filtering services provided by the MASA QoS framework.

5.1 Internet Television

The first application is used to demonstrate a scenario, where multicast can be used to stream the same video simultaneously from one sender to many receivers. The quality of the stream and the policy for performing adaptation in the case of congestion as well as to support receivers connected via heterogeneous transmission links is determined at the sender site.

We used the JMF filter plug-in described in section 3.1 to realize a very simple implementation of the syntactical filter described in section 4.1. Using the RTP protocol for transmission, the sequence numbers of the packets start with zero and increase for subsequent packets for each frame. The filter assumes, that the encoder sends a priority-ordered sequence of packets, where the high priority base layer packets are sent first, followed by packets with lower priority. If we further assume, that reordering does not occur or could be corrected by appropriate mechanisms, the filter algorithm just forwards the first part of the packet sequence for each frame, depending on the quality factor.

For the measurements a video with 1460 frames and a resolution of 384 x 288 pixels with 15 frames/s has been used. The complete video was filtered with different quality factors ranging from 0.1 to 0.95. Figure 3 shows the results for the average Peak Signal to Noise Ratio (PSNR) (averaged over YC_bC_r) compared to the non-filtered WaveVideo encoded video and the average data rate per frame. Obviously, the data rate tends to decrease much faster with smaller quality factors then the PSNR does. Thus, even relatively high filter ratios only lead to a moderate reduction in the image quality.

These measurements demonstrate, that even with this simple filter implementation a reasonable filter behaviour can be achieved. The JAVA implementation was able to filter the test videos in real time on a standard PC.

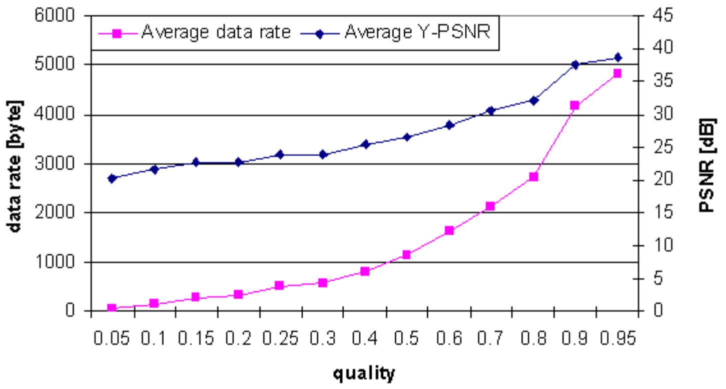


Fig. 3. Averaged PSNR values and data rates for different filter quality settings.

5.2 Video-on-Demand

In the Video-on-Demand scenario, each receiver is applying for a dedicated video stream defined by specific QoS adaptation policies.

We used a very simple implementation of the trading algorithm which is based on user preferences (to build an adaptation path in case QoS violations are detected) and monitoring data from the Mobility Manager (to know when a hand-over occurred and what access network was used). We used the mapping from a single quality parameter and user preferences to derive a set of intervals at the MASA QoS framework level for several categories (frame rate, frame size, visual quality), which we presented in [10]. Based on information about the access network (available from the Mobility Manager) the number of streams to send, the relative importance of the streams and receiver capabilities, we derive a target bandwidth for the sender. This information is available at the trading algorithm, which calculates a set of filter parameters based on local monitoring information (like instantaneous bandwidth), user requirements mapped to intervals and adaptation paths according to [10] and resource availability.

As video source file we used a digitized and WaveVideo compressed scene from the Beverly Hills Cop movie at 352x288 pixel resolution. The original frame rate was 25 fps. In our scenario, the receiver was not interested in receiving more than 10 fps and could not manage more than 1 MBit/s. As frame rate intervals, the receiver specified]8,...,10] fps as desirable,]5,...,8] fps as tolerable, and [2,...,5] fps as minimum acceptable. The Media Controller uses a combi-filter that allows to adapt frame rate (to scale down from 25 to at most 10 fps) and frame quality to match available bandwidth and the preferences.

We conducted the following scenario. First, the sender was connected to the fixed network. After sending 500 frames, the fixed network connection was terminated, which resulted in a hand-over to a WaveLAN base station. After another 500 frames, a hand-over to a new base station was performed that shows better signal quality. Frame rate was judged more important than visual quality. As a consequence, the trader tried to maintain the 10 fps while maintaining the 1 MBit/s during the first 500 frames (phase one), see figure 4. The Media Controller derived the following settings: $r_{src} = 25$, $r_{dest} = 10$ to drop the HH, HL and LH subband of the highest layer ($r_l = 1$). Note that the original stream was based on I- and Delta frames so the instantaneous frame rate varied between 5 and 12 fps. The average frame rate of 10 fps was maintained during the first period. The data rate varied between 400 kBit/s and 1 MBit/s. Without invoking the combi filter, the data rate would have been more than 1 MBit/s.

At frame 500, the Mobility Manager detected a handoff to WaveLAN with a low signal quality. As a result, the QoS Broker instructed the Media Controller to reduce the data rate to at most 200 kBit/s and comply to the user preferences. However, this was only possible if the frame rate was also reduced. The Media Controller therefore applied the following settings: $r_{src} = 25$, $r_{dest} = 4$, $r_l = 5$ thus reducing the average frame rate to 4 fps (minimum acceptable interval) and dropping more subbands of all those frames that are to be forwarded. The data rate was thus reduced to values between 180 kBit/s and 90 kBit/s in phase two.

The receiver noticed that the visual quality was bad as a result of its preferences, so he signaled to the sender that visual quality is as important as frame rate and that a new setting should be applied if resource situation allows this. After frame 1000, a hand-over to a better access point occurred. The QoS Broker instructed the Media Manager/Controller to use the following settings: $r_{src} = 25$, $r_{dest} = 6$ thus increasing frame rate to 6 fps (tolerable interval). As upper data rate limit the trader calculated 500 kBit/s, which the Media Controller used to invoke the combi filter with $r_l = 3$. More subbands were added to increase the visual quality and the data rate varied between 200 kBit/s and 500 kBit/s in phase three.

We can draw the following conclusions from this example. If the original stream is encoded at a variable bitrate, applying a frame rate, luminance quality, chrominance quality filter will not change the variable nature of the stream. Thus the post filter stream also shows a variable data rate (see figure 4). If a constant data rate is desired the dynamic rate shaping filter with priorities should be used [7]. An integration of media management and mobility management enables wireless devices to re-act and adapt to QoS fluctuations on the wireless link by observing user preferences. Finally, adaptation should be co-ordinated among the different entities: resource management, media management, profile management and the coordination unit itself. The adaptation has to be coordinated with the peer because a change in the media stream will influence remote resource consumption. Integration of local, network and remote resources is necessary to provide a full end-to-end QoS aware service.

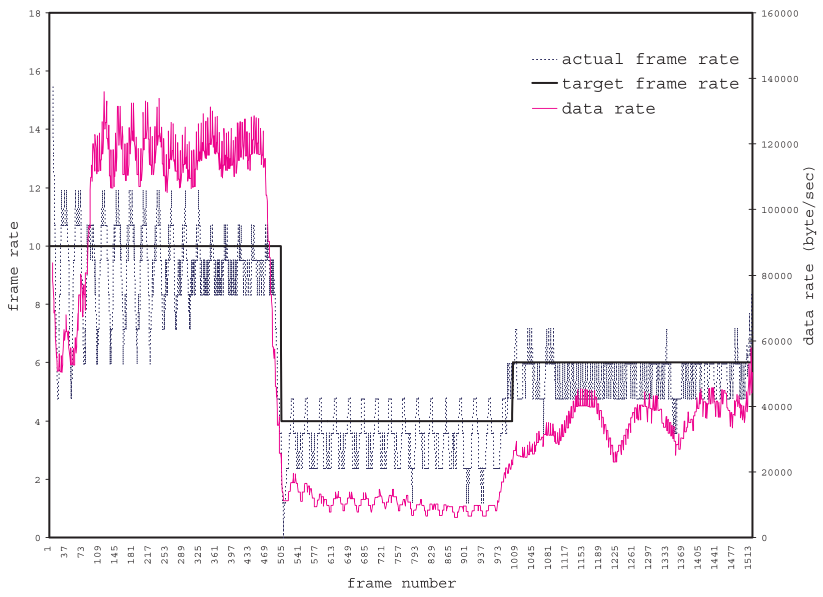


Fig. 4. Frame rate and data rate for the Beverly Hills Cop movie based on combi filter.

6 Conclusions

In this paper we have proposed a flexible filtering strategy supporting adaptive transmission of video streams in a heterogeneous mobile environment. By a combination of syntactical and semantical filter algorithms at filter nodes inside the transmission path, adaptation to varying network situations concerning QoS policies could be achieved. The filter network was implemented within the MASA QoS framework, which allows for a comprehensive management of multimedia streams. The functionality and efficiency of the filtering approach have been demonstrated by first measurements in two example scenarios supporting unicast and multicast applications.

We are currently investigating mechanisms to support multiple different QoS policies in a multicast scenario at the same time. Therefore, we are working on the definition of algorithms to map aggregated subtree requirements to packet drop decisions. Also, enhanced feedback mechanisms are under development, which allow for an optimized parametrization of the codec itself.

References

1. C. Schremmer and C. Kuhmünch and W. Effelsberg. Layered Wavelet Coding for Video. In *Workshop on Packet Video*, 2001.
2. E. Amir and S. McCanne and M. Verterli. A layered DCT coder for Internet Video. In *Proceedings of the IEEE Conference on Image Processing ICIP'96*, September 2001.
3. F. Raspall and C. Kuhmünch and A. Banchs and F. Pelizza and S. Sallent. Study of packet dropping policies on layered video. In *Workshop on Packet Video*, 2001.
4. G. Fankhauser, M. Dasen, N. Weiler, B. Plattner, and B. Stiller. WaveVideo - An Integrated Approach to Adaptive Wireless Video. *ACM Monet, Special Issue on Adaptive Mobile Networking and Computing*, 4(4), 1999.
5. H. Hartenstein, A. Kassler, M. Krautgärtner, A. Schrader. High Quality Mobile Communication. In *Proceedings of the KIVS'2001 Conference*, 2001.
6. J.D. Solomon. *Mobile IP.* Prentice Hall, 1998.
7. A. Kassler and A. Neubeck. Self Learning Video Filters for Wavelet Coded Videostreams. *Proceedings of the Int. Conf. of Image Processing (ICIP2000)*, September 2000.
8. A. Kassler, A. Neubeck, and P. Schulthess. Filtering Wavelet based Video Streams for Wireless Interworking. *Proc. of the ICME*, July 2000.
9. A. Kassler, A. Neubeck, and P. Schulthess. Classification and Evaluation of Filters for Wavelet Coded Videostreams. *Signal Processing: Image Communication*, 16(8):795–807, May 2001.
10. A. Kassler and P. Schulthess. An End-to-End Quality of Service Management Architecture for Wireless ATM networks. *Proc. of the HICSS 32*, January 1999.
11. Steven MacCanne and Van Jacobson. Receiver-driven layered multicast. In *Proceedings of ACM SIGCOMM'96*, August 2001.
12. Sun. *The Java Media Framework Version 2.0 API.* http://java.sun.com/products-/java-media/jmf.

On the Utility of FEC Mechanisms for Audio Applications

Eitan Altman[1,2], Chadi Barakat[1], and Víctor M. Ramos R.[3,⋆]

[1] INRIA Sophia Antipolis, France
{altman, cbarakat}@sophia.inria.fr
[2] CESIMO, Universidad de Los Andes, Venezuela
[3] Institut Eurécom, 06904 Sophia Antipolis, France
Victor.Ramos@eurecom.fr

Abstract. FEC mechanisms have been proposed to recover from packet losses, and hence to improve the perceived quality in audio applications. Recently, it has been shown in [1] that the redundancy added by a FEC scheme increases the congestion of the network and deteriorates the audio quality instead of improving it. In this work we show via a simple queuing analysis that the impact of FEC on the audio quality is not always negative and that we can get better quality in some scenarios. In particular, we show that FEC is beneficial when a small number of flows implement it or when the audio applications have some particular utility functions. We derive conditions on when to get a gain in quality as well as bounds on the maximum gain that we can obtain.

1 Introduction

Forward Error Correction (FEC) is now considered as the most appropriate solution for the recovery from packet losses in audio (or more generally multimedia) applications [2]. This technique consists in transmitting, together with the original audio packets, some redundant information that can be used at the audio receiver to reconstruct any packet lost within the network. Generally, the reconstruction of the lost audio packets should improve the intelligibility of the received audio signal. The redundant information is constructed at the audio source using the original packets and it is sent to the destination in separate packets or piggybacked in subsequent ones. The main advantage of FEC, which makes it very suitable for audio applications, is that packet losses can be reconstructed on runtime without any retransmission from the side of the source. This runtime reconstruction reduces the variations of the end-to-end delay since the receiver is no longer needed to wait until the source retransmits the lost packets. The variation on the end–to–end delay, often called jitter, is an important factor in assessing the quality of an audio transmission. Audio receivers need to implement playout buffers in order to absorb these variations and play audio packets at a regular rate [3]. An important jitter will then result in an important buffering time, in an important end-to-end delay, and hence in a poor quality[1].

⋆ The author is also an associate professor at the Universidad Autónoma Metropolitana in Mexico City.

[1] An audio conversation is considered to be interactive if the two-way end-to-end delay is less than 250ms, including media coding and decoding, network transit and playout buffering [4].

This advantage of FEC has motivated many developers of audio applications to incorporate it into their tools (e.g., Freephone [5] and Rat [6]). Different FEC schemes have been proposed in the literature for this purpose: parity and block erasure codes, convolutional codes, interleaving, multiple description codes, etc. We will focus in this paper on a simple FEC scheme that has been standardized [7] by the IETF (Internet Engineering Task Force) and that has been implemented in many recent audio tools. The scheme simply consists in adding a redundant copy of the original audio packet to the tail of the subsequent one. If it happens that an audio packet is lost while crossing the network and that the following packet is correctly received, the lost packet can be reconstructed from the redundant information contained in the following one. Figure 1 depicts a particular case of this simple FEC scheme, where the offset (ϕ) between the original packet and its copy is equal to 1. Usually, the redundant information is obtained by coding the original packets with a low bit-rate codec. For example, an original audio packet can be coded with PCM and its copy with GSM [8] or LPC [9].

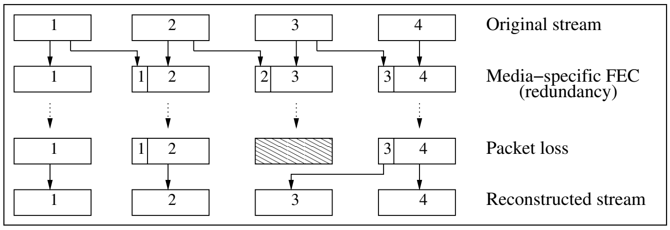


Fig. 1. Simple FEC mechanism where packet $n + 1$ carries redundant information on packet n.

Different works have tried to improve the performance of this simple FEC scheme. Some authors propose to increase the offset (ϕ) between the original packet and its copy [10,11]. Their argument is that the loss process of packets in the Internet is bursty [12,13,14,15,16] and hence, by moving away the redundancy from the original packet, we increase the probability that the redundant copy of an audio packet is correctly received when the original packet is lost. Other authors propose to add multiple redundant copies of a packet in multiple subsequent ones [10,17]. The authors in [18] show that, by adding to an audio packet a redundant copy computed from a block of some preceding packets, we get a better audio quality than when adding to the audio packet a redundant copy computed from a single preceding one. In [18], the authors proposed different ways to group packets in blocks. But, all these works ignore an important fact, that the addition of redundancy increases the transmission rate of the audio sources which may increase the load of the network and hence the loss probability of packets. The study of the performance of a FEC scheme under a constant loss rate leads certainly to an improvement in quality since the number of packets played at the receiver is larger. However, the quality may deteriorate instead of improving if the loss rate of audio packets considerably increases due to the addition of FEC. In this later case, the addition of FEC will not compensate the increase in the loss rate caused by FEC.

Recently, it has been shown in [1] via a queuing analysis that the simple FEC scheme we outlined above does not lead to an improvement of audio quality. The authors in [1] considered a single bottleneck node for the network and focused on the case when the buffer size in the bottleneck router is only dedicated to the audio flow (or to an aggregate of audio flows implementing the same FEC scheme and sharing the same bottleneck). The assumptions made in [1] hold when all flows in the network implement FEC, or when a round-robin scheduler with per-flow queuing is used. Under these assumptions, the authors in [1] show that even for the infinite-offset case ($\phi \to \infty$) which forms an upper bound on the audio quality, adding FEC according to this simple FEC scheme leads always to a deterioration of quality caused by an important increase in network load.

In this work we address the questions of how and where this simple FEC scheme, which we recall is implemented in many audio tools as Freephone and Rat, leads to an improvement in quality. The negative result given in [1] holds in the case when all the flows in the network add FEC, or when the audio flow has its own buffer in network routers. It also holds with the particular utility function the authors considered. A utility function indicates the variation of the audio quality at the receiver as a function of the transmission rate. The authors in [1] considered a linear utility function; they supposed that the more the user receives data, the better is the quality and that the increase in quality for a certain amount of redundancy is the same for any value of the transmission rate. In fact, the quality of an audio transmission is quite a subjective measure and it is known to be non-linear [19]. Moreover, the audio source may use different code rates for FEC which will result in different qualities for the same value of the transmission rate. We look here at cases where the assumptions in [1] are not satisfied and we try to understand why this simple FEC scheme improves the audio quality in some scenarios. We use some queuing models for this purpose. Our findings in this paper can be summarized as follows:

– With a linear utility function as the one used in [1], the addition of FEC leads to an improvement in quality if the (total) rate of the flow(s) adding FEC is small compared to the total rate of the other flows sharing the same bottleneck and not adding FEC. The addition of FEC in this case does not lead to an important increase in the loss rate which explains this improvement. We start to lose in quality when the (total) rate of the flow(s) using FEC increases.
– The audio quality is always an increasing function of the offset between the original packet and its copy.
– In the case when all flows are adding FEC, which forms the worst case where the addition of FEC has the biggest impact on the load of the network, it is possible to obtain a gain in quality for some particular utility functions. The utility function must increase with the amount of FEC faster than the linear one, and higher increase rates are required for small amounts of FEC. In some words, to gain in quality, a small amount of FEC must lead to approximately the same quality as the original audio packet.

The remainder of this paper is organized as follows. In Section 2 we investigate the case of a single audio flow sharing the bottleneck with an exogenous traffic not using FEC. In Section 3 we study the performance of the FEC scheme described above for different utility functions. We conclude this work in Section 4. Note that although we are focusing on audio flows, our results on FEC are valid for any other kind of multimedia application.

2 Multiplexing and FEC Performance

2.1 The Model

Consider the case of an audio flow implementing FEC and sharing a bottleneck router with some other flows not implementing FEC. We look at the other flows as a single exogenous flow of constant rate and of packet size exponentially distributed. The latter choice can be justified by the mixture of a large number of flows from different sources and of different packet sizes. Let $1/\mu$ denote the average transmission time at the bottleneck of a packet from the exogenous flow. This time is independent of the amount of FEC added to the audio flow. We consider that the original audio packets (before the addition of FEC) have a fixed length and we denote by $1/\mu_0$ their average transmission time at the output interface of the bottleneck router.

Let us suppose that packets (audio + exogenous) arrive at the bottleneck router according to a Poisson process of constant rate λ. Suppose also that audio packets arrive at the bottleneck according to a Poisson process. This latter assumption can be justified by the fact that audio packets cross multiple routers before arriving at the bottleneck, so that their inter-arrival times can be approximated by an exponential distribution. Let $\beta \in [0,1]$ denote the fraction of arriving packets belonging to the audio flow; this quantity represents the probability that a packet arriving at the bottleneck is of audio type. Suppose finally that the bottleneck router implements the classical Drop Tail policy and has a buffer of size K packets (packet in service included). Packets from different flows share the K places of the buffer and are served in a FIFO (First-In First-Out) fashion. The system can be then considered as an $M/G/1/K$ queuing system where packets arrive according to a Poisson process and where service times (or transmission times in our settings) are independent and identically distributed. This system can be then solved using some known results from queuing theory [20,21]. Our main objective is to find an expression for the audio quality at the destination as a function of the different system parameters as well as the amount of FEC added to the original packets by the audio source.

2.2 The Analysis

Suppose first that the audio flow does not implement FEC. We look at the audio quality at the moments at which packets would arrive at the destination. We take a value equal to 1 as the quality obtained when the audio packet is correctly received, and 0 as the quality when the packet is lost in the network. The average audio quality during the conversation is equal to $Q = 1 - \pi$, where π denote the stationary probability that a packet is dropped in an $M/G/1/K$ system. This probability is equal to $\pi = \frac{1+(\rho-1)f}{1+\rho f}$, where ρ is the total system load (or the total traffic intensity) given by $\rho = \lambda(\frac{\beta}{\mu_0} + \frac{1-\beta}{\mu})$, and f is the $K-2$ th coefficient of the Taylor series of a complex function $G(s)$ defined as $G(s) = (B^*(\lambda(1-s)) - s)^{-1}$. $B^*(s)$ is the Laplace Stieltjes transform of the service time distribution [21]. In our case,

$$B^*(s) = \int_0^\infty b(t)e^{-st}dt = \beta e^{-s/\mu_0} + (1-\beta)\mu/(\mu+s), \quad \text{for } Re(s) \geq 0\,.$$

The coefficient f can be computed by developing the Taylor series of the function $G(s)$ with some mathematical symbolic software [2]. It can also be calculated using the theorem of residues as follows:

$$f = \frac{1}{(K-2)!} \left. \frac{d^{K-2}G(s)}{ds^{K-2}} \right|_{s=0} = \frac{1}{2\pi i} \oint_{D_r} G(s) \frac{ds}{s^{K-1}}\,,$$

where D_r is any circle in the complex plane with center 0 and with radius chosen small enough so that the circle does not contain any pole of the function $G(s)$.

Now, the addition of FEC to the audio flow according to the FEC scheme we described in Section 1 increases the transmission time of audio packets at the output interface of the bottleneck router. This increases the load of the system which changes the stationary probabilities. Let $\alpha \in [0,1]$ denote the ratio of the volume of FEC at the tail of a packet and the volume of the original packet. The new transmission time of audio packets becomes $\frac{(1+\alpha)}{\mu_0}$, and the new system load becomes

$$\rho_\alpha = \lambda\Big(\frac{\beta(1+\alpha)}{\mu_0} + \frac{(1-\beta)}{\mu}\Big)\,. \tag{1}$$

In the same way we can compute the new transform of the transmission time, the new coefficient f, and the new drop probability of an audio packet (it is the same for exogenous packets given that the arrival processes of both flows are Poisson). Henceforth, when we add an index α to a function, we mean the new value of the function after the addition of an amount α of FEC. The quality after the addition of FEC becomes

$$Q_\alpha^\phi = (1-\pi_\alpha) + U(\alpha)\pi_\alpha(1-\pi_\alpha^\phi)\,. \tag{2}$$

The first term corresponds to the quality obtained when the original audio packet is correctly received. The second term corresponds to the quality obtained when the redundant copy is correctly received and the original packet is lost. $U(\alpha)$ indicates how much quality we get from an amount α of FEC. The quantity π_α^ϕ indicates the probability that the packet carrying the redundancy is dropped given that the original packet is also dropped. ϕ represents the offset (in number of audio packets) between the original packet and the one containing its copy. In this section we will only consider the case of a utility function $U(\alpha) = \alpha$ similar to the one studied in [1]. We keep the study of the impact of other utility functions until Section 3.

The exact computation of Q_α^ϕ requires the computation of π_α^ϕ. This latter function is quite difficult to calculate given the multiplexing of packets from both flows at the bottleneck. We must summarize over all the possible numbers of non-audio packets inserted between audio packets. What we can do instead is to find bounds on this probability and thus bounds on the quality. From [1], the probability that a packet is lost

[2] As Maple (http://www.maplesoft.com) or Mathematica (http://www.wolfram.com).

given that the n-th previous packet is lost is a decreasing function of n and it converges to π_α when $n \to \infty$. We can write $\pi_\alpha \leq \pi_\alpha^\phi \leq \pi_\alpha^0$, with π_α^0 being the probability that a packet (from any flow) is lost given that the previous packet is also lost. This gives us the following two bounds on the quality: $Q_\alpha^0 \leq Q_\alpha^\phi \leq Q_\alpha$, where

$$Q_\alpha^0 = (1-\pi_\alpha) + \alpha\pi_\alpha(1-\pi_\alpha^0)\,, \tag{3}$$

$$Q_\alpha = (1-\pi_\alpha)(1+\alpha\pi_\alpha)\,. \tag{4}$$

We use these two bounds to study how the audio quality varies for different amounts of FEC and for different intensities of audio traffic. We are sure that if we gain in Q_α^0 (lose in Q_α), we will gain (lose) in quality for any offset. Our main objective here is to show how the quality varies with FEC for different values of β. It has been shown in [1] that we always lose in quality for $\beta = 1$ (i.e., when the audio flow occupies 100% of the bandwidth at the bottleneck). All that we still need to do is to find the expression for the lower bound on the quality which can be found from the expression of π_α^0.

Theorem 1 *π_α^0 is given by $1 + \frac{B_\alpha^*(\lambda)-1}{\rho_\alpha}$, with*

$$B_\alpha^*(\lambda) = \beta e^{-\lambda(1+\alpha)/\mu_0}$$

and ρ_α given by equation (1).

Proof: Consider a general $M/G/1/K$ queuing system. We have to compute the probability that a packet (say 1) is dropped given that the previous packet (say 0) is also dropped. Let $a(t) = \lambda e^{-\lambda t}$ be the distribution of time intervals between arrivals (of packets from both flows), and let $b(t)$ be the distribution of service times. Let $r(t)$ be the distribution of the residual time for the packet in service when packet 0 arrives (there is certainly a packet in service since packet 0 is supposed to be dropped). Using the results in [20], we write $r(t) = \frac{1-B(t)}{\sigma}$. $B(t)$ is the cumulative distribution function of the service time and σ is the average service time. In our case,

$$B(t) = \beta 1\{t \geq (1+\alpha)/\mu_0\} + (1-\beta)(1-e^{-\mu t}),$$

and $\sigma = \rho_\alpha/\lambda$. The probability π_α^0 is no other than

$$\pi_\alpha^0 = \int_0^\infty \frac{1-B(t)}{\sigma}(1-e^{-\lambda t})dt\,.$$

This is the probability that the inter-arrival time between packet 0 and packet 1 is less than the residual time of the packet in service, and we summarize over all the possible values of the residual service time. With a simple calculation on this expression and by using the new values of the load intensity and the Laplace Stieltjes Transform of the service time distribution after the addition of FEC, we can prove the theorem.

2.3 Numerical Results

We solve numerically the model for the two bounds on the audio quality (Eq. 3 and 4). We set K=10 packets and λ=10000 packets/s. Without loss of generality, we set μ_0=μ.

We consider four values of ρ: 0.5, 0.8, 1, and 1.5. For every value of ρ, we plot the audio quality as a function of β and α. Recall that β is the fraction of audio packets and α is the amount of FEC. Figure 2 shows the results.

We conclude from the above figures that it is possible to obtain a gain with the simple FEC scheme we are studying. This requires that the intensity of the audio flow is small compared to the intensity of the other flows not implementing FEC. The gain diminishes as long as the intensity of the flows implementing FEC increases. It disappears when most of the flows start to implement FEC. This means that a FEC scheme with a simple linear utility function is not a viable mechanism. The gain that we may obtain in some cases is the result of the fact that the exogenous flows are not adding FEC and then they are not so aggressive as audio flows.

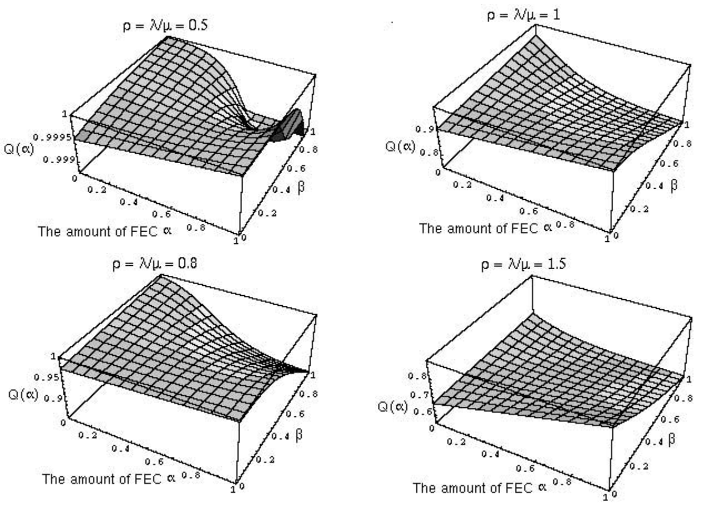


Fig. 2. Audio quality for an $M/G/1/K$ queue with two flows: the audio flow and the exogenous flow. β represents the probability that an arriving packet belongs to the audio flow. We see clearly how when $\beta \to 0$, Q_α^ϕ starts having an increasing behavior, and this gain becomes more important as ρ increases.

3 Utility Functions and FEC Performance

We seek now for a FEC mechanism able to improve the quality in the worst case when all flows in the network implement FEC. Suppose that the audio flow (or an aggregate of audio flows) uses alone the bottleneck resources ($\beta = 1$). The negative results obtained in [1] are due to the linear utility function adopted in the analysis. Adding an amount of FEC α increases the drop probability of an audio packet, which reduces the first term in the right-hand side of (2) more than it increases the second term. To get a gain, the second term must increase faster than the decrease in the first term. This can be achieved if the utility function increases faster than linearly as a function of α.

Indeed, it has been shown in [19] that multimedia applications have different utility functions than a simple linear one. These functions are typically non-linear. They are convex around zero and concave after a certain rate (between 0 and 1, with 1 being the rate that gives a utility function equal to one). Multimedia applications, and audio applications in particular, have strong delay constraints so that the quality deteriorates sharply when the transmission rate falls below a certain value. This kind of utility functions can be very useful for FEC mechanisms since the reconstruction of a packet from a copy of volume $\alpha < 1$ may give approximately the same quality as when the original packet is correctly received. We obtain a gain in quality when the redundant information we add to the original packet is small so that it does not contribute to a big increase in loss probability π, and at the same time, if reconstructed in case of the loss of the original packet, it gives a quality close to 1. Such behavior can be also obtained by coding FEC with a lower-rate codec as GSM [8]. Analytically speaking, a utility function leads to an improvement of quality if for $\alpha < 1$, we have

$$Q_\alpha^\phi = (1 - \pi_\alpha) + U(\alpha)\pi_\alpha(1 - \pi_\alpha^\phi) > (1 - \pi)\,,$$

with π being the stationary drop probability before the addition of FEC.

3.1 Some Bounds on Quality Improvement

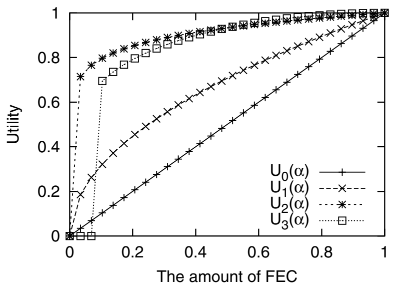
Again, we use here the bounds on the quality $Q_\alpha^0 \leq Q_\alpha^\phi \leq Q_\alpha$, with

$$\begin{aligned} Q_\alpha^0 &= (1 - \pi_\alpha) + U(\alpha)\pi_\alpha(1 - \pi_\alpha^0)\,, \\ Q_\alpha &= (1 - \pi_\alpha)(1 + U(\alpha)\pi_\alpha) \end{aligned}$$

A utility function that improves the lower bound improves the quality for any value of ϕ. A utility function that does not improve the upper bound will not lead to an improvement of quality whatever is the value of ϕ. Using the upper bound, we can find the maximum quality that this simple FEC scheme can give and this is for the best utility function. Indeed, the best utility function is one that jumps directly to one just after 0. This could be subjectively justified by using redundant packets coded at very small rates, as LPC or GSM. A very small amount of FEC ($\alpha \simeq 0$) that does not change the load of the network (i.e., that does not change π), will then lead to the same quality as the original audio packet. The question that one may ask here is: "why to send large original packets in this case, given that we are able to obtain the same quality with small packets?" The important processing time required by low-rate codes could be the answer to this question. We are not addressing this issue here, and we will only focus on the calculation of an upper bound for the FEC scheme we are studying. Let Q^{max} be the maximum quality that we could obtain, thus $Q^{max} \simeq (1 - \pi) + \pi(1 - \pi) = 1 - \pi^2$.

This Q^{max} has to be compared to the quality $(1 - \pi)$ we get in the absence of FEC. Given that Q^{max} is larger than $(1 - \pi)$, we conclude that we can always find a utility function and an offset between original packets and redundancies so as to gain in quality. Note that we are not considering the impact of the coding and decoding delays on the audio quality. The impact of these delays will be the subject of a future work. We also conclude from our analysis here that the FEC scheme we are studying cannot improve the quality by more than a factor of π. This means that the maximum gain in quality we could obtain is 100% and this gain is an increasing function of the network load. For example, for a network that drops 1% of packets, we cannot improve the quality by more than 1%, and for a network that drops 10% of packets we can get an improvement up to 10%.

Without loss of generality, we consider the family of utility functions that jump from zero to 1 at a value α_0. We denote such functions by $U_{\alpha_0}(\alpha)$. These are the utility functions of the so called hard real-time applications. We also consider the upper bound on the quality (an infinite offset). When increasing the amount of FEC with such applications from 0 to α_0, the quality deteriorates since its equal to $(1 - \pi_\alpha)$. When we cross α_0, the quality jumps from $(1 - \pi_{\alpha_0})$ to $(1 - \pi^2_{\alpha_0})$ and it resumes then its decrease with α. For such applications, the FEC scheme improves the quality if $\pi^2_{\alpha_0} < \pi$ and the maximum gain that we could obtain is a factor of $\frac{(\pi - \pi_{\alpha_0})}{(1-\pi)}$. This maximum gain corresponds to an amount of FEC slightly larger than α_0. It is not clear how the gain varies as a function of network load. But, what we can say here is that the FEC scheme behaves better with functions having a small α_0. After a certain threshold on α_0, the above condition becomes unsatisfied and it becomes impossible to gain in quality.



$$U_0(\alpha) = \alpha$$
$$U_1(\alpha) = \sqrt{\alpha}$$
$$U_2(\alpha) = \alpha^{\frac{1}{10}}$$
$$U_3(\alpha) = u(\alpha - \alpha_0)\left(\frac{1-\cos\pi\alpha}{2}\right)^{\frac{1}{10}}$$

Fig. 3. Possible utility functions for rate adaptive applications.

3.2 Some Numerical Results

We give in Figure 3 some possible utility functions[3] that could serve to our needs, and that are similar in their form to the utility functions proposed in [19]. In Figure 3 $U_3(\alpha)$ is plotted with $\alpha_0 = 0.1$.

We solve the model numerically for the two bounds on the quality. We calculate first the stationary distribution of the model for different values of α and ρ. We set K to 20 and λ to 10000 packets/sec. Then, for the the different utility functions in Figure 3, we plot the upper and lower bounds on the quality (Q_α and Q^0_α). Figure 4 shows plots for the lower bound and Figure 5 shows plots for the upper bound. The top four plots were

[3] The function $u(\alpha)$ is the step unit function. It is equal to 1 if $\alpha > 0$, and is equal to zero otherwise. α_0 represents the initial value giving a significant quality.

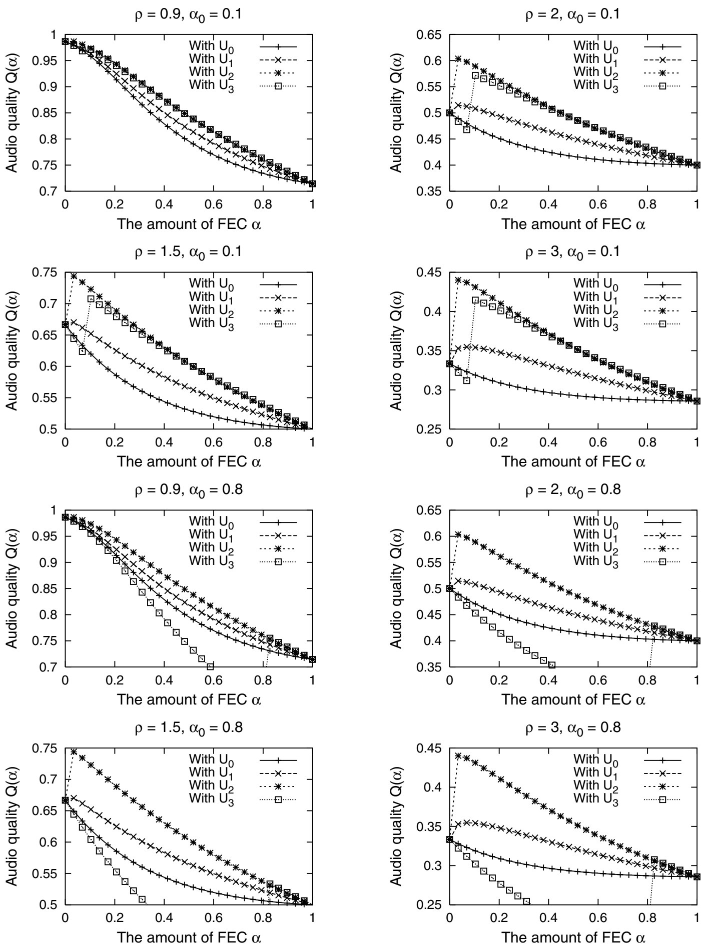


Fig. 4. Lower bound for audio quality with $K = 20$, $\alpha_0 = 0.1$ (top) and $\alpha_0 = 0.8$ (bottom).

obtained with $\alpha_0 = 0.1$ and the four bottom plots with $\alpha_0 = 0.8$ in both figures. We see clearly how the jump in the utility function results in a jump in quality and how this jump leads sometimes to better quality than that at α_0 and sometimes not. We also see how the case $U(\alpha) = \alpha$ does not present any improvement in quality.

4 Conclusions

We showed in this paper that a simple FEC scheme as the one proposed by the IETF and implemented in some audio tools may lead to better performance in two cases. The first case is when the audio flow has a small rate compared to the exogenous traffic. The second case is when the utility function of the audio application presents an important

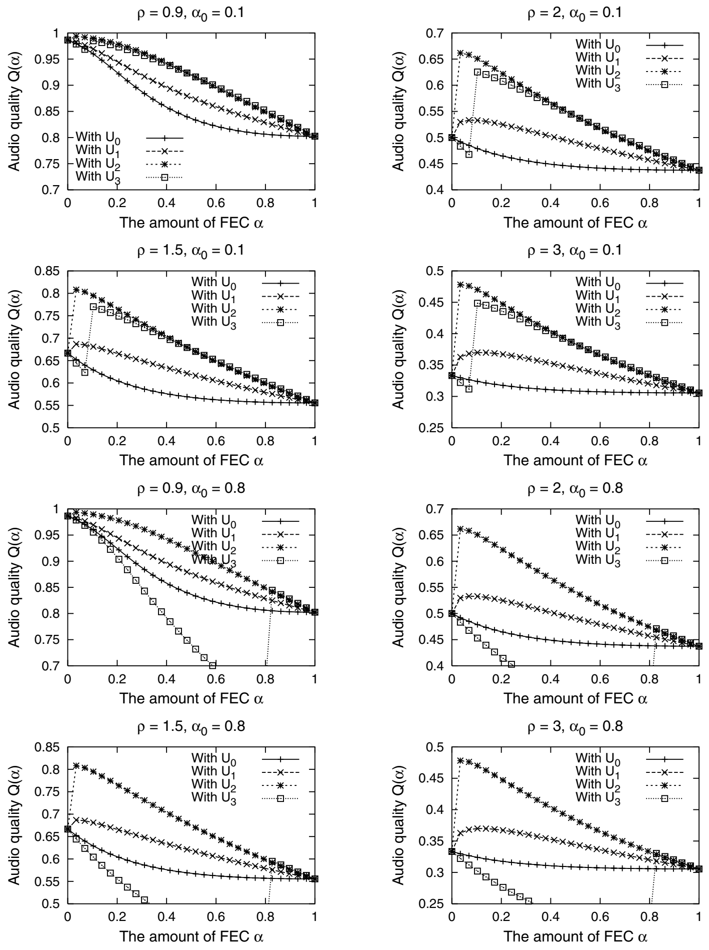


Fig. 5. Upper bound for audio quality with $K = 20$, $\alpha_0 = 0.1$ (top) and $\alpha_0 = 0.8$ (bottom).

jump at small transmission rates. We gave conditions on where the FEC scheme can improve the audio quality.

Although we found some regions where the FEC scheme can behave well, we believe that this scheme is not the appropriate solution for improving the quality of audio applications. In the current Internet, this scheme is profiting from the fact that most of the other flows are not implementing FEC. This will not be the case when all flows start to add FEC to their packets. There is also a problem with the mechanism in case of applications with different utility functions than linear. We found that we get a gain when a small amount of redundancy gives the same performance as the big original packet. It seems intuitive here to reduce the volume of original packets to reduce the drop probability and to gain in quality instead of adding FEC that does not improve the performance by no more than 100%. There is no need to send long packets if we are able to get good quality with small ones.

We believe that the main problem with this kind of mechanisms is that the redundant information is constructed at the source using one packet and so the destination has only two choices: either receive the original packet or receive its copy. Better performance could be obtained if we give the receiver more choices by constructing at the source the redundancy carried by a packet from a block of audio packets. This is what we will investigate in the future.

References

1. Altman, E., Barakat, C., Ramos, V.M.: Queueing analysis of simple FEC schemes for IP telephony. Proc. IEEE Infocom (2001)
2. Perkins, C., Hodson, O., Hardman, V.: A survey of packet loss recovery for streaming audio. IEEE Network (1998)
3. Rosenberg, J., Qiu, L., Schulzrinne, H.: Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet. Proc. IEEE INFOCOM (2000)
4. Perkins, C., Hodson, O.: Options for repair streaming media. RFC 2354 (1998)
5. García, A.V., Fosse-Parisis, S.: (FreePhone audio tool) High-Speed Networking Group, INRIA Sophia Antipolis.
6. Mice Project, T.: (RAT: Robust Audio Tool) Multimedia Integrated Conferencing for European Researchers, University College London.
7. Perkins, C., Kouvelas, I., Hodson, O., Hardman, V.: RTP payload for redundant audio data. RFC 2198 (1997)
8. Scourias, J.: (Overview of the global system for mobile communications) Univ. of Waterloo.
9. Tremain, T.E.: The government standard linear predictive coding algorithm: Lpc-10. Speech Technology **1** (1982) 40–49
10. Bolot, J.C., Fosse-Parisis, S., Towsley, D.: Adaptive FEC-based error control for interactive audio in the Internet. Proc. IEEE Infocom (1999)
11. Kouvelas, I., Hodson, O., Hardman, V., Crowcroft, J.: Redundancy control in real-time Internet audio conferencing. Proc. of AVSPN (1997)
12. Altman, E., Jean-Marie, A.: Loss probabilities for messages with redundant packets feeding a finite buffer. IEEE Journal of Selected Areas in Communications **16** (1998) 779–787
13. Cidon, I., Khamisy, A., Sidi, M.: Analysis of packet loss process in high-speed networks. IEEE Transactions on Information Theory **IT-39** (1993) 98–108
14. Hellal, O.A., Altman, E., Jean-Marie, A., Kurkova, I.: On loss probabilities in presence of redundant packets and several traffic sources. Performance Evaluation **36-37** (1999) 486–518
15. Bolot, J.C., Crépin, H., García, A.V.: Analysis of audio packet loss in the Internet. NOSSDAV (1995)
16. Salamatian, M.R.: Transmission Multimédia Fiable Sur Internet. PhD thesis, Université Paris Sud UFR Scientifique d'Orsay (2000)
17. García, A.V.: Mécanismes de Contrôle pour la Transmission de l'Audio sur l'Internet. PhD thesis, Université de Nice Sophia-Antipolis (1996)
18. Figueiredo, D.R., de Souza e Silva, E.: Efficient mechanisms for recovering voice packets in the Internet. Globecom (1999)
19. Shenker, S.: Fundamental design issues for the future Internet. IEEE Journal on Selected Areas in Communications **13 (7)** (1995) 1176–1188
20. Kleinrock, L.: Queueing systems. John Wiley, New York (1976)
21. Cohen, J.W.: The Single Server Queue. North-Holland (1969)

A Call Admission Control Method for Supporting Telephony Sessions in a Best Effort IP Network

István Szabó

Ericsson Research, Traffic Analysis and Network Performance Lab,
H-1300 Budapest 3., P.O. Box 107, Hungary
Tel.: +36 1 4377627 Fax:+36 1 4377767
Istvan.Szabo@ericsson.com

Abstract. An actively studied research area of the past couple of years is the design of connection admission control techniques for IP networks. This paper proposes a new, end–to–end measurement based approach for call admission control in IP telephony gateways. The method relies on measuring the quality experienced by ongoing live sessions: it collects packet loss and delay statistics about the sessions active between peer telephony gateways. This leads to very fast admission decision compared to existing methods, and also eliminates the signalling load and the maintenance of associated flow state in the core routers. The paper contributes to answering the fundamental question of what sort of QoS guarantees can be provided by a core–stateless QoS provisioning architecture, and it also investigates the effect of flows subjected to end–to–end measurement based admission control on legacy TCP traffic.

1 Introduction

Both incumbent telecommunication operators and emerging new mobile and fixed telephony service providers are willing to offer telephony services over their IP network. An important property of voice flows is that they tolerate certain loss. For example, the error concealment unit of Adaptive Multi-rate (AMR) speech codec can provide undistorted speech up to 1% frame error rate, and it can provide acceptable speech quality up to 10% frame error rate [1]. IP networks today are in most cases equipped with best effort routers, and it will take significant amount of time and money to upgrade all of them with features to enable service differentiation. In order to provide acceptable quality voice communication, the best effort service paradigm is not sufficient. Some method is needed to block new call arrivals if the capacity limit of the network is reached. Many approaches have been proposed in the past couple of years for service differentiation and resource allocation in IP networks. These methods represent different trade–off between efficient utilisation of bandwidth and implementation complexity. On the long road towards full–fledged, QoS capable, multiservice IP networks, the deployment of the recently proposed end–to–end measurement based admission control methods [2, 3] might represent a sound migration step. They can assure certain service guarantees for sessions, while their implementation complexity and deployment cost is very limited, since they do not require any upgrade of the routers in the network core.

The application scenario is depicted in Fig. 1. The access network can be of basically any kind; it can be a traditional PSTN network or a UMTS Terrestrial Radio Access Network. It is further assumed that some sort of end–to–end call/session level signaling protocol (H.323, SIP, DSS1, ISUP, BICC, or their appropriate combination) is used to control the calls. When a call establishment message hits the gateway, the gateway makes a call admission control decision to check that the IP network has enough capacity to accommodate the new call. If the decision is positive, the call/session level signalling proceeds towards the remote end, if not the call is blocked. The goal here is to develop and evaluate a call admission control method, which can be used by IP telephony gateways, and requires no involvement from the core routers.

Traditional resource provisioning methods rely on some sort of signalling protocol [4] to indicate resource requests to the routers in the core of the network. Such a request travels back and forth between the telephony gateways, which delays the connection establishment at least by a round trip time. In this paper a solution is evaluated which minimizes the call establishment delay by not requiring any inquiry of the network state on a per call basis.

When it comes to deploying a new solution over best effort IP core networks, it is very important to investigate the interaction with legacy network traffic, especially to find the effect on TCP sessions, which represent the biggest portion of the traffic in IP networks today.

The contribution of this paper is twofold: a new end–to–end measurement based call admission solution is proposed, which can be used over pure best effort core networks, and extensive simulation results are presented, which reveals not only the quality of service level provided for the voice calls subjected to the call admission control, but also investigates the effect of admitted sessions on competing TCP traffic, showing that TCP can be adversely effected, and care needs to be taken before deploying end–to–end measurement based call admission control in operational IP networks.

The remainder of this paper is organised as follows. Section 2 presents the call admission control method. The next section is concerned with simulation based performance evaluation. Related work is summarised in Section 4, and finally, conclusions are drawn.

2 The Call Admission Control Method

In the network depicted in Fig. 1 telephony calls are traveling through the core network between pairs of IP Telephony Gateways. There are hundreds of simultaneous calls between any two IP Telephony Gateways. The base of the call admission control method is that the gateways collect aggregate statistics (such as packet loss ratio, delay) about the ongoing calls, and regularly share this information with the peer gateway. When a new call arrives at the gateway, it compares the latest available statistics with the target value set for the calls, and accepts the call if the performance indicators are below a preset threshold. The target values should be set in a way that the tolerable delay is not exceeded by the packets of admitted voice sessions, and the packet loss is below the limit, which can be compensated by the voice codec. As far as the start–up phase is concerned, when no statistics is available, a reasonable assumption is that a core network can handle thousands of simultaneous calls, so the first few calls can for sure be admitted without compromising the performance.

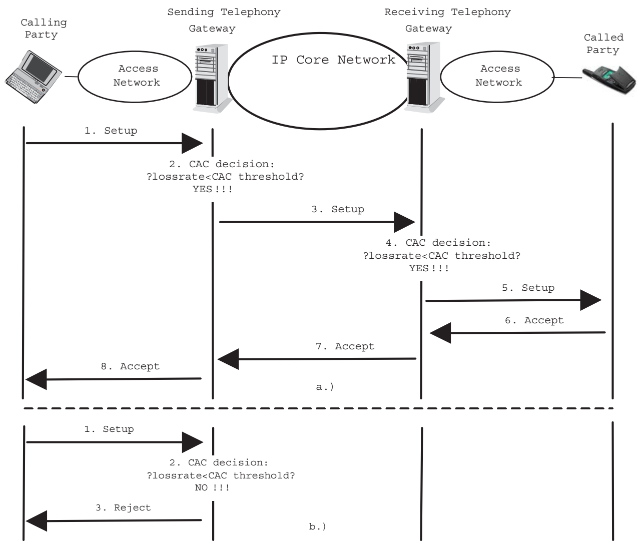

Calling Party
Sending Telephony Gateway
Receiving Telephony Gateway
Called Party
Access Network
IP Core Network
Access Network
1. Setup
2. CAC decision: ?lossrate<CAC threshold? YES!!!
3. Setup
4. CAC decision: ?lossrate<CAC threshold? YES!!!
5. Setup
6. Accept
7. Accept
8. Accept
a.)
1. Setup
2. CAC decision: ?lossrate<CAC threshold? NO !!!
3. Reject
b.)

Fig. 1. Actions for successful (a.) and unsuccessful (b.) connection establishment

The “receiving gateway” is delivered an IP packet encapsulating a voice frame. Before sending the voice frame to the destination access network, the gateway determines which gateway have sent this packet (based on the source IP address in the packet header), and which call the frame belongs to (based on the call identifier included in the packet header). It checks the frame sequence number to calculate the number of packets that have been lost, and updates the packet loss counter corresponding to the sending gateway. It also checks the timestamp to calculate the transmission delay, and to update the delay counter. At the end of each measurement period the receiving gateway sends a control packet to each sending gateway to inform them about the loss and delay statistics of their calls. If the voice frames are carried in RTP frames, the control information can be conveyed by the control protocol RTCP [5]. Note however, that RTCP establishes a control stream for each individual voice call, so using RTCP means that as many control packets needs to be exchanged between the gateway pairs in each measurement interval, as many calls are active in that particular interval. In the current implementation one control packet is sent reporting aggregate statistics about all the calls which are active between the gateway pairs. This approach very much limits the overhead resulting from the measurement reports.

After receiving a voice frame from the access network, the ‘‘sending gateway” determines which call the packet belongs to, and it increases the frame sequence number of that particular call, and the gateway encapsulates the voice frame into an IP packet. The encapsulation can for example be done using RTP framing. The only requirement from the framing protocol is to provide sequence numbering, and to allow the identification of individual calls. The destination address field of the IP header is set to the address of the receiving gateway. The receiving gateway can for example be looked up in a static routing table.

Upon receiving a control packet from the remote gateway, the state variables “lossrate” and “actdelay”, which stores the loss and delay suffered by the voice packets of the gateway are updated, typically by calculating a weighted average of the latest measurement result, and the previous value.

Fig. 1 depicts the sequence of actions taking place when an end user initiates a new voice call. Note that the message names appearing in the figure don’t imply the use of any particular call control protocol. The end–to–end measurement based admission control method can work together with all existing call control signalling solutions. A connection setup message travels through the access network of the calling party, and arrives at the “sending” IP telephony gateway. The gateway derives the address of the “receiving” gateway from the address of the called party. It compares the actual value of loss and delay measurements towards the “receiving” gateway with the preset thresholds (“losstarget” & “delaytarget”). If the actual values are below the thresholds, the new call is accepted, and the connection establishment message is forwarded to the remote gateway (Step 3 of Fig 1.a). If any of the current figures is above the respective threshold, the call is rejected (Step 3 of Fig. 1.b). When making the call admission control decision (Step 4), the “receiving” gateway uses the measurement reports submitted by the “sending” gateway.

As it is clear from the presentation above, the computational overhead of the method is very small. The method puts burden only on the edge routers (IP telephony gateways). A counter needs to be maintained for each active flow to be able to determine the packet loss. Three more counters are needed, one to calculate the number of packets arriving in one measurement interval, and a second one to accumulate the number of packet losses, and a third one to accumulate the delay values.

3 Performance Evaluation

This section presents numerical results obtained by simulation. The results characterize the effect of admission controlled non–adaptive flows on the throughput of competing TCP traffic, and also show the quality experienced by individual voice calls. The goal of the simulations is to understand whether admission controlled traffic can fairly co–exist with legacy TCP flows while acceptable voice quality is also maintained.

The call admission control method is implemented in ns-2 [6]. All the simulations presented in this paper were run for 1000 simulated seconds, and repeated 4 times with different random seeds. Data collected during the first 300 seconds were discarded. The results shown are averages over the 4 runs. The propagation delay of the links is set to 15ms. The bottleneck buffer sizes are scaled with the link capacity as proposed in [7]: it is set to 15·linkrate packets, where the linkrate is measured in Mb/s.

Since the method presented here is targeted primarily for supporting voice users, an important aspect of the simulation is the voice traffic model. A widely accepted On/Off voice model is used. The sources generate 70 bytes long packets, in ON periods, while in OFF periods no packet is sent. This packet size corresponds to the standard, uncompressed RTP/UDP/IP framing of voice data. The duration of ON and OFF periods are exponentially distributed, the mean length of ON periods is 352 ms, and the mean length of OFF periods is 650 ms, as suggested in [8]. The voice call arrival process is Poisson, and the call holding time is exponentially distributed with a mean of 90 s.

TCP flows use Reno version of the congestion control algorithm, which is widely deployed. TCPs are fed by an FTP application simulating the download of a large file, which means that TCP has data to send during its entire lifetime. TCP generates 1000 bytes long packets. Each TCP source and sink is connected to the first hop router via a dedicated 100 Mb/s link. The propagation delay of the links between the sources and the first hop router is set to 2 ms, while it is set to a different value for each connection between the first hop router and the sinks. This ensures that the TCP flows have different round trip time.

The literature uses the so–called loss–load curves to demonstrate the performance of a particular measurement based admission control technique. This curve shows the average loss suffered by the flows subjected to the admission control. Such an average value is quite meaningless when it comes to packet switched compressed voice communication. A state of the art voice codec can conceal quite high loss rates, so for voice communication the relevant question is that given a CAC method, how many percent of the individual voice flows exceed a certain loss threshold. In this paper figures show the fraction of individual voice flows, which suffered more than 10% loss. With a typical voice codec, this value can be regarded as the upper bound for providing acceptable voice quality.

The first experiment is run with a single bottleneck link interconnecting two routers. The bottleneck link is shared by varying number of TCP flows and admission controlled voice traffic. The utilisation of the bottleneck link is measured, and the fraction of bandwidth received by the admission controlled traffic is calculated. The feedback period is set to 1s. Fig. 2 depicts the bandwidth share of the admission controlled flows. The value of 0.5 would represent fair sharing between the TCP flows and the admission controlled traffic. The average flow inter–arrival time of the voice traffic is scaled with the bandwidth of the bottleneck link. It is set to 1.5/linkrate seconds, where the linkrate is measured in units of Mb/s. The loss threshold is set to 0.1 and the delay threshold is set to 150 ms, which corresponds to acceptable one–way delay for real–time voice communication. (Subjective tests reported in Annex B of [9] say that the communication quality is acceptable if the one–way mouth to ear transmission delay stays below 400–500 ms. Based on this figure we assume in this paper that about 150–250 ms can be consumed from the delay budget in the IP core network.)

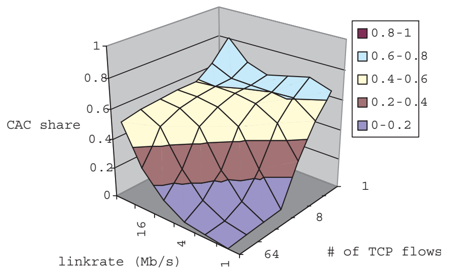


Fig. 2. Share of admission controlled flows

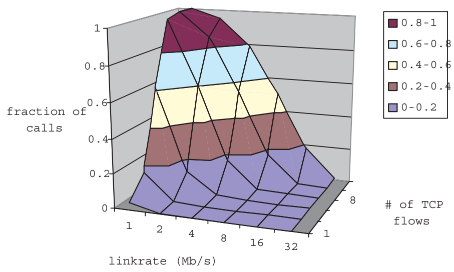


Fig. 3. Fraction of calls with >10% loss

We can observe that the admission control solution leads to a reasonably fair bandwidth sharing (value between 0.4–0.6) over a large operating region. TCPs are much more aggressive in fighting for bandwidth if a small bottleneck link (1–4 Mb/s) is used by a large number of TCP flows. On the other hand, the admission controlled traffic gets more than its fair share if it competes with only one or two TCP flows. However, even in this extreme case, the TCP traffic is not completely expelled. The utilisation of the bottleneck is very high. Its minimum is 74% when there is only a single TCP flow competing with the voice traffic over a 32 Mb/s link. The average utilisation over all 42 simulated cases is 96%.

To assess the quality of service guarantees provided for the voice traffic the blocking probability and the fraction of individual voice calls, which suffered more than 10% packet loss are measured, and the 10^{-2} percentile of the delay of the voice packets is also calculated. The blocking probability calculation confirms the observation above that in case of a small bottleneck link and many simultaneous TCP flows, the admission controlled traffic is hit very hardly (~100% blocking).

Fig. 3 shows the loss statistics. The graph demonstrates that not just the overall bandwidth is shared fairly, but also the quality of service experienced by individual voice sessions is satisfactory. The packet loss ratio of individual calls is below 10% for most values of the simulated parameter range. Once again, one can note that problems start to occur when many TCP flows are competing against the voice traffic over a small link. The worst case queuing delay in this configuration is around 120 ms, which is measured when the small buffer configured for a 1–2 Mb/s link is almost constantly kept occupied by the large number of active TCP flows. This worse case delay value also indicates that all the blocked calls are rejected due to high loss values, and never due to excessive queuing delay.

To learn more about the degradation of the communication quality, two more parameters were calculated. The first one is the maximum length of loss bursts that is the maximum number of consecutive packets that has been lost from a particular voice session. This parameter indicates[1] the length of annoying periods of speech clipping. The second parameter is the maximum length of speech burst, which is the maximum number of consecutive speech packets that has been delivered without packet loss. This value measures the length of completely undistorted speech periods. Table 1 summarises the result. A speech burst of 8918 packets represents approximately 3 minutes of talking.

Table 1. Length of undistorted speech and loss periods

Linkrate	# of TCPs	Loss burst	Speech burst
2 Mb/s	64	9	53
8 Mb/s	8	7	409
32 Mb/s	4	4	8918

The second experiment concentrates on the effect of different loss thresholds used by the call admission control algorithm. These simulations are also run with a single bottleneck link, which has 8 Mb/s capacity. The delay measurement results are not taken into account when making the admission control decision. Fig. 4 depicts the bandwidth share of the admission controlled traffic. The minimum utilisation of the bottleneck is 95%, while the average utilisation over all the simulation runs is higher than 99%. If there is no call admission control, the non–responsive voice traffic dominates the link. By lowering the call admission control threshold, smaller and smaller number of TCP flow is enough to suppress the admission controlled traffic. These curves show that end–to–end measurement based call admission control can ensure fair sharing of the link capacity only in a limited region. One set of simulation is run to check the effect of Random Early Discard (RED). RED is configured as in [7]: the buffer capacity is set to 120, the minimum threshold is 15 packets, the maximum threshold is 50 packets, the maximum drop probability is 0.1 and the weight used in the queue size calculation is 0.002. If there are many parallel TCP sessions, RED buffer management results in noticeably higher blocking probability and lower share for the voice sessions compared to the case when using the same CAC threshold but droptail buffer management. The difference in the bandwidth share is about 8-14%.

[1] Note that this parameter can not be used as a direct measure of the length of the speech clipping period because the error concealment unit of the voice coder can significantly alleviate the user–perceived effect of packet loss.

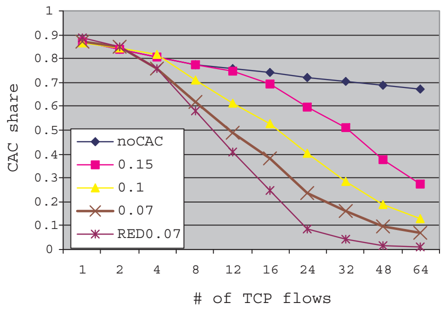


Fig. 4. Share of admission controlled flows

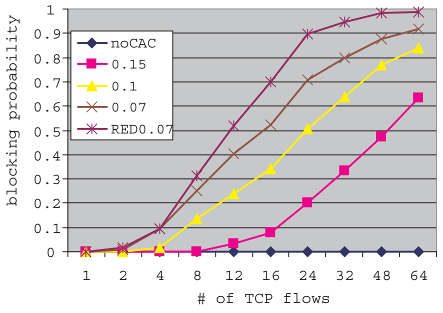


Fig. 5. Blocking probability

Fig. 5 depicts the blocking probabilities. The amount of traffic generated by the voice flows is proportional to the number of flows admitted into the network, therefore it is not surprising that the blocking probability curves basically mirror the corresponding bandwidth share curves.

Fig. 6 shows the loss performance of individual voice flows, and Fig. 7 depicts the voice packet delay. Setting the CAC threshold to the value of 0.07 results in far the best performance. Even if there are 64 competing TCP flows, only 8% of the voice calls suffer more packet loss than the critical 10%. The voice QoS degrades significantly with the other four settings. The worst performance is delivered if the voice calls are not regulated at all. Fig. 4 and 6 demonstrate the benefit of applying even this very simple call admission control technique. The proposed method can definitely prevent the worst possible scenario, when non–adaptive voice flows flood the bottleneck link, and they not only expel TCP traffic from the network, but render also the voice flows to unacceptable performance. It is interesting to see the effect of RED buffer management on the packet loss suffered by individual voice flows. RED starts to make a very big difference when the number of TCP flows is high. Random drops help TCPs avoiding synchronized shrinking of the congestion windows, which result in larger portion of the buffer being allocated by TCP packets, so higher chance for a voice packet to arrive to the buffer when the queue length is measured to be above the maximum threshold.

All 4 CAC thresholds yield to roughly the same delay performance if the number of TCP flows is below 12 and the queue management method is tail drop. In case of RED queue management, the delay performance is better because in a lightly loaded system RED drops more packets than tail drop, but there is not enough TCP flow to benefit from random drops, so the buffer occupancy, and consequently the voice packet delay is smaller. In this region the number of TCP flows is not really effecting the delay, the curves are flat. As the number of TCP flows goes beyond 12, the different loss thresholds yield to significantly different delay performance. Stricter loss threshold means less and less voice flows in the system when the number of competing TCPs increase. This means that the portion of large TCP packets in the system increases therefore the voice

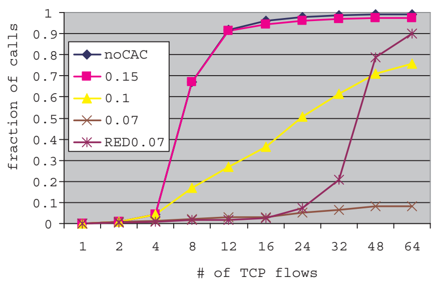


Fig. 6. Fraction of calls with >10% loss

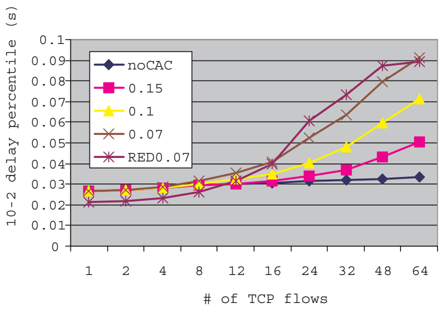


Fig. 7. Voice packet delay

packets are queued behind higher and higher number of 1000 bytes long TCP packets, which explains the difference between the delay curves.

The third experiment tries to answer the question whether higher intensity of feedback messages makes the admission controlled traffic more competitive compared to TCP. These simulations are also run with a single bottleneck link, which has 8 Mb/s capacity. 3 simulations are run with a delay threshold of 80ms. The voice packet delay is averaged over the feedback interval, and new calls are blocked if the average delay exceeds the 80 ms delay threshold. Fig. 8 shows the bandwidth share of the admission controlled traffic and Fig. 9 depicts the loss performance. (3 curves are incomplete due to blocking of all calls.)

When the background load is light (1-16 TCP flows), there is a benefit of configuring more frequent measurement reports. (0.1s or 0.5s instead of 1.0s). In this case the reason for call blocking is mainly packet loss, which is confirmed by the fact that the performance of the method is the same even if no delay threshold is configured (mea1.0_nod). As a reference, the graphs also show the measurement results for the case when the call admission control method is switched off (noCAC). The utilisation of the bottleneck link is above 99% if averaged over all the simulation runs. Increasing the frequency of measurement reports by a factor of 10 results in a smaller bandwidth share for the voice traffic. (The difference is approximately 3%.) As a reward, the loss performance gets better that is the fraction of calls delivering unacceptable performance decreases by ∼10%.

On the other end of the scale when there are a lot of TCP flows, blocking also occurs due to measuring higher average delay than the delay threshold. The method with more frequent feedback is less conservative. The benefit is however questionable since the per–flow loss values are very bad. The delay for 99% of the voice packets is well below the delay threshold in all settings.

Fig. 8 gives the impression that switching on the use of delay measurements in the call admission control method makes the method more conservative when it comes to fighting for bandwidth. The aim of the final set of simulations is to justify this observation. A new network topology, depicted in Fig. 10, is used where the capacity of the two backbone links (node2–node3, node3–node6) is 8 Mb/s, the capacity of the other links is 100 Mb/s. The admission control method

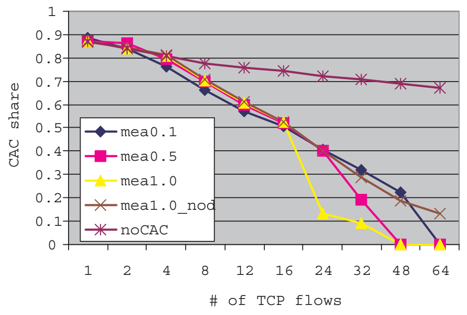


Fig. 8. Share of admission controlled flows

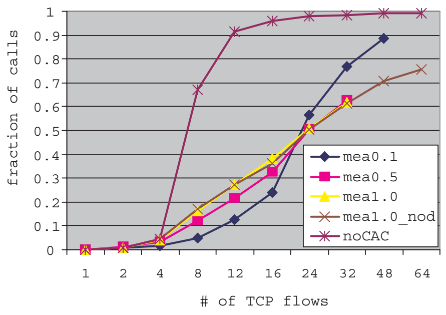


Fig. 9. Fraction of calls with >10% loss

operates between node1 and node4. The link between node2 and node3 is a bottleneck during the whole simulation, because TCP flows generate traffic from node0 to node5. Between 650 and 800 second of the simulation time, there is also TCP traffic generated between node8 and node7. The number of simultaneous TCPs is equal on both bottlenecks. Three simulations are presented in Table 2. The first two simulations are run with a single bottleneck, the feedback interval is 0.1s, but the delay and loss thresholds are different. The third one is run with two bottlenecks without using the delay measurements in the CAC algorithm.

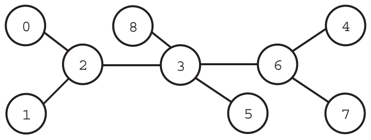


Fig. 10. Network configuration with two bottlenecks

Comparing the first and second column of Table 2 reveals that a smaller delay threshold leads to a very abrupt change in the bandwidth share of the admission controlled traffic, when the number of competing TCP flows increases. Once there is enough TCP in the network to raise the bottleneck buffer occupancy above a limit corresponding to the delay threshold of the admission control algorithm, there will be no new call admitted. Setting the delay threshold to a value, which is smaller than the maximum queuing delay basically means that voice packets can only be queued up to a buffer limit corresponding to the preset delay threshold, while TCP packets can exploit the whole buffer. This inevitably leads to expelling of the voice traffic earlier compared to the case when no delay threshold is set in the CAC method. Column 3 confirms this observation: if no delay threshold is configured, the admission controlled traffic is more competitive.

Table 2. Summary of the simulations with different delay thresholds

	1 bottleneck, meas=0.1s delay<0.05s, loss<0.1			**1 bottleneck, meas=0.1s delay<0.08s, loss<0.07**			**2 bottlenecks, meas=1s no delay th., loss<0.1**		
# TCPs	**CAC share**	**>10%**	**Block**	**CAC share**	**>10%**	**Block**	**CAC share**	**>10%**	**Block**
1	0.888	0.002	0.004	0.854	0.002	0.016	0.897	0.001	0
8	0.443	0.054	0.447	0.588	0.015	0.299	0.762	0.053	0.043
16	0.341	0.219	0.598	0.403	0.03	0.492	0.6	0.244	0.244
24	0	N/A	1	0.289	0.134	0.647	0.465	0.367	0.371
32	0	N/A	1	0.222	0.229	0.723	0.386	0.457	0.467

4 Related Work

Elek et al.[2] proposed a solution, where the host probes the network before sending data. During the probe interval the destination host measures the packet loss rate, and reports it back to the initiator. If the reported packet loss ratio is below a threshold, the new session is admitted.

Bianchi et al.[3] introduces an analytical model to evaluate the performance of this probe stream based solution, but the model works only for constant bit rate sessions, while Breslau et al. [10] present a thorough simulation based performance evaluation of the method. [10] mentions three important drawbacks of the tested solution: (1)The long connection establishment delay induced by the probing period. (2)The overhead of probing streams, and (3)The possibility of network underutilization in case of high offered traffic due to the fact that the probe load is so high that each probing session experiences packet loss above the preset threshold, so none of them will finally be admitted.

The solution proposed in this paper eliminates all three problems by relying on measuring ongoing, live sessions, plus it also incorporates delay measurements into the decision making. The possible interaction of end–to-end measurement based admission control techniques with legacy TCP traffic has not yet been studied in detail.

5 Conclusions and Future Work

A fast and lightweight call admission control method was presented, which is tailored to the needs of IP Telephony Gateways. It is fast because there is no need to inquire the core network for resources on a per call basis. It is lightweight because the core routers don't need to be upgraded at all.

The method was studied with extensive simulations. The goal of the investigation was to understand the effect of TCP flows on the admission controlled traffic. The results presented here demonstrate that end–to–end measurement based call admission control is a viable option for IP telephony gateways, and it represents a very reasonable trade–off between the complexity of the method, and the offered performance guarantees. The most important findings of the paper are as follows.

Increasing the number of TCP flows inevitably results in complete expelling of the admission controlled traffic, simply because TCP is not that concerned with the packet loss ratio, and can work nicely with much higher packet loss ratio than the one which can be tolerated by the voice traffic. The method however has a very definite benefit: It blocks all those voice calls, which would anyway be useless because of the very high packet loss ratio, and in this way ensures much higher throughput for TCP traffic.

In the range, where the number of TCP flows present in the network is modest, that is the resulting packet loss ratio is not more than the one which can be tolerated by the voice traffic, the method is extremely beneficial for two reasons: (1)It protects TCP traffic from non–responsive flows, and result in an approximately fair share of the bandwidth between the two traffic classes. (2) It ensures the required packet loss ratio for the voice flows.

The method can ensure that the delay bound of the voice packets is not exceeded. Taking however delay statistics also into account in the admission control decision makes the method more conservative when fighting for bandwidth against TCP flows.

The presented method can best be used in IP telephony gateways. To design a distributed measurement architecture, which would make the solution applicable in endpoints without requiring them to send probe streams, is a challenging item for future work.

References

1. E. Ekudden, S. Bruhn, and P. Sörqvist. The adaptive multi–rate speech coder - the new flexible world–standard for voice compression. In *XVII. World Telecommunications Congress*, Great Britain, May 2000.
2. V. Elek, G. Karlsson, and R. Rönngren. Admission control based on end–to–end measurements. In *IEEE Infocom*, Israel, April 2000.
3. G. Bianchi, A. Capone, and C. Petrioli. Throughput analysis of end-to-end measurement–based admission control in IP. In *IEEE Infocom*, Israel, April 2000.
4. IETF RFC 2205. *Resource ReServation Protocol (RSVP)*, 1997.
5. IETF RFC 1889. *RTP: A Transport Protocol for Real–Time Applications*, 1996.
6. *UCB/LBNL/VINT Network Simulator ns (version 2) can be found at: http://www-mash.cs.berkeley.edu/ns/ns.html.*
7. S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation–based congestion control for unicast applications. In *ACM Sigcomm*, Sweden, August 2000.
8. S. Deng. Traffic characteristics of packet voice. In *IEEE International Conference on Communications*, USA, June 1995.
9. ITU-T Recommendation G.114. *General Characteristics of International Telephone Connections and International Telephone Circuits — One–way Transmission Time*, 1996.
10. L. Breslau, E.W̃. Knigthly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *ACM Sigcomm*, Sweden, August 2000.

Integrated Admission Control for Streaming and Elastic Traffic

N. Benameur, S. Ben Fredj, F. Delcoigne,
S. Oueslati-Boulahia, and J.W. Roberts[*]

France Télécom R&D, Issy Les Moulineaux, France.
{nabil.benameur,slim.benfredj,franck.delcoigne,
sara.boulahia,james.roberts}@francetelecom.com

Abstract. It is proposed to apply an integrated admission control scheme to both streaming flows and elastic flows. It is assumed that streaming flow packets are served with priority in network queues so that admission control ensures minimal delay for audio and video applications while preserving the throughput of document transfers. An implicit measurement-based implementation is proposed where admissibility is based on an estimation of the bandwidth a new elastic flow would acquire. An analytical fluid flow model provides insight and guides the choice of admission thresholds. Detailed packet level simulations of TCP and UDP connections show that the proposed algorithms work satisfactorily in the range of admission thresholds predicted by the fluid model.

Key words: Admission control, quality of service, streaming traffic, elastic traffic, flow aware networking.

1 Introduction

Applications expected to produce the bulk of traffic in the future multiservice Internet can be broadly categorized as streaming or elastic according to the nature of the flows they produce. Streaming flows are produced by audio and video applications. Elastic flows, on the other hand, result from the transfer of digital documents (Web pages, files, MP3 tracks,...) using a transport protocol like TCP. The Diffserv network architecture goes someway towards meeting the distinct quality of service requirements of these two types of traffic. However, it remains unclear how this architecture can be used by network providers to offer useful services with meaningful end-to-end guarantees. In this paper we argue that an essential network function is missing: this function is admission control applied to both streaming and elastic flows.

Admission control has been studied mainly in the context of streaming traffic with new flows being rejected if the addition of their traffic would lead to quality degradation for ongoing flows. It has traditionally relied on signalling and

[*] Partial funding for the research reported here was provided by the European Union framework programme ITEA through the project RTIPA.

explicit resource reservation. The use of admission control for elastic traffic has been proposed more recently with the objective of avoiding unnecessary loss of performance and efficiency in case of demand overload [1,2,3]. In contrast with previous studies, which have addressed the issue of admission control either for streaming traffic alone or for elastic traffic alone, in this paper we envisage an integrated network where streaming and elastic flows share the same links and admission control applies to both. Packets of streaming flows have priority in network queues to minimize their delay while elastic flows dynamically share the remaining bandwidth.

The envisaged admission control is *implicit*: new flows are identified "on the fly" and if a flow must be rejected, its packets are simply discarded. Avoiding the need for signalling and explicit resource reservation considerably simplifies implementation which can be introduced progressively with minimal need for standardization. We propose a measurement-based implementation where admissibility is based on an estimation of the bandwidth which a new elastic flow would acquire. We investigate two alternative criteria for deriving this: the measured rate of a permanent "phantom" connection, and the current packet loss rate experienced by traffic on the considered link or path. The corresponding admission control algorithms were introduced in an earlier work and tested by simulation assuming all traffic is elastic [4]. In this paper, we reexamine these algorithms with the objective of investigating the impact of streaming traffic on their efficiency. To gain insight into their performance in this context and to guide our choice of admission threshold, we develop a fluid model of statistical bandwidth sharing accounting for both classes of flow.

The rest of the paper is organized as follows. We first discuss in Section 2 traffic characteristics of elastic and streaming flows. Section 3 describes the integrated network architecture we envisage. Section 4 presents the proposed fluid model and its analysis. Results of the simulation of the proposed admission control algorithms are presented in Section 5. Conclusions are drawn in Section 6.

2 Flow Level Traffic Characterization

For the purposes of traffic engineering and control it is most convenient to characterize demand at flow level [5]. For present purposes we assume that a flow can be characterized as a sequence of packets having the same identifier transmitted with an inter-packet interval smaller than some threshold (10 or 20 seconds, say). Flows may be broadly divided into two categories: elastic and streaming.

2.1 Elastic Flows

Elastic flows correspond to the transfer of digital documents (Web page, file, MP3 track,...) and adapt their rate to available capacity by means of a transport protocol like TCP. Such document transfers currently constitute the majority of Internet traffic. An elastic flow is simply characterized by the size of the document to be transferred. The time required to transfer the document depends on the number of ongoing flows on all routes and constitutes the main performance criterion for an elastic flow. Equivalently, we will consider the throughput achieved for a document transfer.

A simple model of elastic traffic at flow level is to assume flows arrive according to a Poisson process and have a size drawn independently from a certain distribution. From the result of measurements, a candidate distribution would have a heavy tail: most flows are very small (so-called "mice") while the majority of traffic is contained in very long flows (so-called "elephants"). It has recently been shown that results derived for this simple Poisson arrivals model are in fact valid under much more general and realistic traffic assumptions [6]. Traffic demand in bits/s is equal to the product of arrival rate and mean flow size.

2.2 Streaming Flows

Streaming flows are produced by audio and video applications and have an intrinsic rate which must be preserved by the network. The essential traffic characteristics of streaming flows are their duration and rate.

We assume streaming flows arrive according to a Poisson process and last for a duration drawn independently from a certain distribution. Most audio and video applications have variable rate. However, the impact of rate variations on realized performance is minimized with the use of rate envelope multiplexing which is the assumption made in this paper. In rate envelope multiplexing the combined input rate is maintained below the link service rate [7]. Streaming traffic demand is equal to the product of arrival rate, duration and mean flow bit rate.

2.3 Integrating Streaming and Elastic Flows

We assume streaming flows are handled in network nodes using non-preemptive priority queuing. Giving priority to streaming packets ensures maximal responsiveness for the underlying audio and video applications. The impact of non-priority traffic on streaming flow performance has been shown to be negligible in a rather precise sense [8].

Elastic traffic shares the remaining bandwidth unused by streaming traffic by means of the congestion control algorithms of TCP. The generally assumed objective is that bandwidth is shared as fairly as possible between contending flows although the degree of fairness achieved by TCP is variable depending on many factors including the connection round trip time and the maximum window size of the different connections.

3 Implicit Integrated Admission Control

We consider an integrated network where streaming and elastic flows share the same links and admission control applies to both. We successively describe the salient features of our admission control scheme and the algorithms used.

3.1 The Need for Admission Control

The objective of admission control in an integrated network is twofold. First, it must maintain the time integrity of streaming flows by ensuring low delay and jitter and second, it must preserve the throughput of elastic flows in the event of overload. The model of statistical bandwidth sharing for elastic traffic introduced in [9] shows that performance is excellent as long as demand does not exceed capacity. Hence, the main objective of applying admission control to elastic traffic is to avoid congestion collapse in the event of overload manifested by an ever increasing number of flows in progress with per-flow throughput tending to zero.

3.2 Implicit Admission Control

Implementing an admission control mechanism on a per-flow basis raises serious scalability issues. The very high flow arrival rate on any network link and the small size of most flows require an *implicit* admission control procedure that avoids per flow signalling and resource reservation. The admission control scheme must estimate in real time the ability of a link or path to accept a new flow. Our approach is to rely on a minimal description of each flow coupled with a measurement-based admissibility condition.

We propose to identify new flows on the fly and to reject them, when necessary, using existing protocol semantics at transport layer and above. A list of flows in progress is maintained and the flow identity of all arriving packets is systematically compared with this list. The flow identity would be determined from certain fields in the packet header (e.g., source and destination addresses and port numbers, the flow identity field of IPv6). If a packet belongs to an existing flow it is forwarded; if not, either the new flow is added to the list if it is accepted, or the packet is simply discarded. The discard of the first packets of a flow is generally sufficient signal to the source that resources are unavailable. Flows would be erased from the table whenever the time since their last packet exceeds a certain threshold.

Alternative implicit admission control procedures have been proposed in [1,2]. In these procedures, flow identification is only performed for TCP connections which are delimited by the initial three-way handshake. In the interests of generality, we prefer in our approach to avoid depending on the detection of SYN and SYN/ACK packets.

3.3 Measurement-Based Admissibility Condition

The admissibility condition determines whether or not it is possible to accept a new flow. Since the rationale behind applying admission control to elastic traffic is to ensure admitted flows maintain useful throughput, the admissibility condition is logically related to this performance parameter. We propose therefore to estimate the *available bandwidth*, i.e. the bandwidth a new elastic flow would acquire by sharing capacity fairly with the flows already in progress. Given the inherent tolerance of elastic flows to rate fluctuations, a rough estimate of available bandwidth is sufficient. A significant advantage of this approach is that it applies equally to a single link and to a network path. This is a desirable feature when admission control decisions are performed in the edge routers of an MPLS domain, say, as envisaged in [3].

Streaming flows would be admitted in the conditions of rate envelope multiplexing at a rather low load (assuming the majority of traffic is elastic). Admission control then ensures a negligible probability of input rate exceeding the multiplexer service rate [7]. Measurement-based admission control has been studied in this context and shown to constitute a practical solution avoiding the need to know complex traffic descriptors for each flow [10]. In the present approach, we simply assume the peak rate of streaming flows is guaranteed to be less than a certain maximum value, less than or equal to the available bandwidth threshold used for admission control. Use of a common threshold avoids starvation for the highest rate flows in case of heavy traffic and considerably simplifies implementation since it is unnecessary to signal precise traffic parameters.

Measurement-based admission control is particularly easy in an integrated system where the majority of traffic is elastic. Since packets of streaming flows have priority their loss rate is likely to be negligible when the admissibility condition for an elastic flow (i.e., sufficient available bandwidth) is satisfied. Any imprecision in the algorithm leads to momentary rate reduction for elastic flows but has negligible impact on the performance of admitted streaming flows.

3.4 Bandwidth Estimation Algorithms

We consider two available bandwidth estimation algorithms. The first runs an artificial “phantom” TCP connection, as proposed by Afek et al [11], and measures the bandwidth it currently receives simply by averaging the short term rate of acknowledged packets. The second measures the current packet loss rate p and uses the fact that TCP adjusts the sending rate in reaction to packet loss inducing a relationship between p and the achieved throughput. In practice, it is not straightforward to estimate the loss rate on the link or path in question. One possibility would be to generate a stream of probe packets and to measure their loss rate. In the simulations we measure the loss rate by counting the numbers of offered and lost elastic flow packets.

4 Performance Model

In this section, we present a fluid model of statistical bandwidth sharing on a link integrating both elastic and streaming traffic. The model guides our choice of admission control thresholds and allows an evaluation of the impact on performance of particular traffic characteristics.

4.1 Fluid Flow Model

Consider a single bottleneck link of capacity C bits/s shared by streaming and elastic flows arriving according to Poisson processes of intensity λ_s and λ_e, respectively. Let the mean size of elastic flows be $1/\mu_e$ bits and the mean duration of streaming flows $1/\mu_s$ seconds. We assume streaming flows have constant bit rate d_s. Denote by $\rho_s = \lambda_e/(C\mu_e)$ and $\rho_e = \lambda_s d_s/(C\mu_s)$ the link load induced by streaming traffic and elastic traffic, respectively and write ρ for the overall load $\rho_s + \rho_e$.

The bandwidth of elastic flows varies dynamically depending on the number of flows in progress of both types. For present purposes, we assume the link bandwidth unused by streaming traffic is shared perfectly fairly with instant readjustment whenever new flows begin or existing flows end. Note, however, that detailed packet level simulations of TCP and UDP connections are used in Section 5 to validate the proposed mechanisms.

We assume there is a lower threshold $d_{\min}$ on the acceptable throughput of elastic flows. In other words, any newly arriving flow is rejected if the instantaneous flow throughput would otherwise decrease below the threshold $d_{\min}$. Write,

$$d = \max(d_s, d_{\min}).$$

Let Q_s and Q_e represent the number of ongoing streaming flows and elastic flows, respectively. Feasible occupancy states are such that

$$Q_s d_s + Q_e d_{\min} \leq C - d.$$

The use of a unique admissibility condition for streaming and elastic flows yields equal blocking probabilities for both types and prevents the most tolerant class gaining an unfair advantage in case of heavy traffic.

4.2 Quasi-Stationary Analysis

Let $N_e(i)$ denote the maximum number of elastic flows when there are i ongoing streaming flows:

$$N_e(i) = \left\lfloor \frac{C - i d_s - d + d_{\min}}{d_{\min}} \right\rfloor,$$

and N_s the maximum number of streaming flows in the absence of elastic traffic:

$$N_s = \left\lfloor \frac{C - d + d_s}{d_s} \right\rfloor.$$

Let $\rho_e(i)$ denote the elastic traffic load when i streaming flows are present:

$$\rho_e(i) = \frac{\lambda_e}{\mu_e(C - i d_s)}.$$

Rather than solving this system exactly under Markovian assumptions (this would only lead to a complex algorithmic solution), these quantities are computed below under a quasi-stationary, or QS, assumption: the ratio λ_s/λ_e is assumed small enough so that, in the presence of i ongoing streaming flows, Q_e evolves rapidly with respect to Q_s and attains a stationary regime. The number of elastic flows then behaves like the population of an M/G/1 processor sharing queue of load $\rho_e(i)$ with upper limit $N_e(i)$. In this QS regime we have:

$$\Pr\left[Q_e = j | Q_s = i\right] = \frac{(1-\rho_e(i))\rho_e(i)^j}{1-\rho_e(i)^{N_e(i)+1}}, \ \forall j \le N_e(i). \tag{1}$$

Let B denote the flow blocking probability:

$$B = \sum_i \Pr\left[Q_s = i\right] \frac{(1-\rho_e(i))\rho_e(i)^{N_e(i)}}{1-\rho_e(i)^{N_e(i)+1}}. \tag{2}$$

To compute $\Pr\left[Q_s = i\right]$, note that Q_s behaves like the population of an M/G/∞ system with state dependent arrival rate and maximum size N_s. The arrival rate when $Q_s = i$, denoted $\lambda_s(i)$, is λ_s thinned by the probability a flow is not blocked (i.e., $Q_e < N_e(i)$):

$$\lambda_s(i) \ = \lambda_s \frac{1-\rho_e(i)^{N_e(i)}}{1-\rho_e(i)^{N_e(i)+1}}, \ \forall i < N_s.$$

The distribution of Q_s under the QS assumption is thus,

$$\Pr[Q_s = i] = \Pr\left[Q_s = 0\right] \prod_{k=0}^{i-1} \frac{\lambda_s(k)}{(k+1)\mu_s}, \ \forall i \le N_s. \tag{3}$$

where $\Pr[Q_s = 0]$ is given by the normalization condition $\sum \Pr[Q_s = i] = 1$.

The blocking probability B can thus be derived from (2). To evaluate throughput performance we proceed as follows. The expected response time R of an elastic flow can be deduced using Little's formula with the expected number of elastic flows in progress $E[Q_e]$ derived from the distributions (1) and (3):

$$R = \frac{E[Q_e]}{\lambda_e(1-B)}.$$

Now, it is known that the response time $R(s)$ of a flow of size s in an M/G/1 processor sharing queue is proportional to s [12]. The constant of proportionality can be deduced on remarking that $R = E[R(s)]$, implying $R(s) = R\mu_e s$. Define $\gamma_e = s/R(s)$ to be the (harmonic) mean throughput of a flow of any size [6]:

$$\gamma_e = \frac{\rho_e(1-B)}{E[Q_e]}.$$

It is noteworthy that, under the QS assumption, the above results for the performance parameters of interest B and γ_e are insensitive to the distributions of elastic flow size and streaming flow duration.

4.3 Validity of the QS Assumption

We have simulated the fluid flow system with the objective of validating the analytical model and examining sensitivity with respect to size and duration distributions when the QS assumption is not appropriate. Assumed parameter values are as follows: $C = 10$ Mbit/s, $d_s = 0.0015C$, $1/\mu_e = 200$ Kbits, $1/\mu_s = 60$ s. We consider a class of hyper-exponential distributions for both streaming flow duration and elastic flow size defined as follows:

$$\forall x \geq 0, \quad \Pr[s > x] = \frac{a \exp^{-ax/\sigma} + \exp^{-x/(a\sigma)}}{a+1}$$

where σ is the mean and the parameter a controls the proportions of short and long flows.

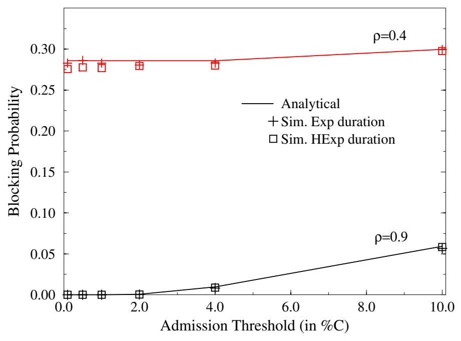


Fig. 1. Blocking probability

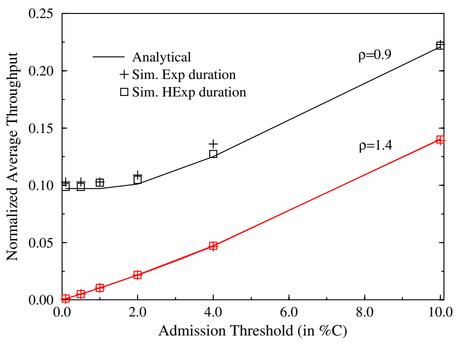


Fig. 2. Normalized throughput

Figures 1 and 2 plot the blocking probability B and the normalized average throughput γ_e/C, respectively, against the admission threshold for $\rho = 0.9$ and $\rho = 1.4$. Elastic flow sizes are drawn from the hyper-exponential distribution with $a = 100$. We show simulation results for two different distributions of streaming flow duration, exponential ($a = 1$) and hyper-exponential ($a = 100$).

We observe that the simulation results are very close to the analytical results in all cases and for all considered duration distributions. We have also validated the approximation for different flow size distributions and for a shorter mean duration of 30 s. The results confirm that the analytical fluid model constitutes a good approximation under rather general and realistic traffic patterns.

4.4 Setting the Admissibility Threshold

It may readily be verified that when $\rho < 1$ the blocking probability B tends rapidly to zero as the threshold decreases below $0.02C$, e.g., a threshold of $0.01C$ yields $B < 0.001$ for $\rho \leq 0.95$. On the other hand, when $\rho > 1$ we have $B \approx (\rho - 1)/\rho$ whenever the threshold is less than $0.02C$.

When $\rho < 1$, the normalized average throughput is virtually independent of the threshold (and equal to $(1 - \rho)$) as soon as the threshold is smaller than

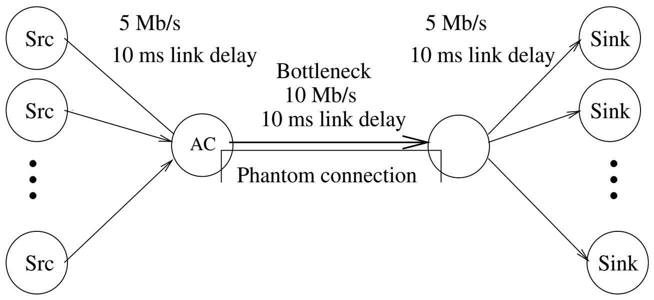


Fig. 3. Simulated network topology

$0.02C$. In overload, on the other hand, throughput is roughly proportional to the threshold since the link is almost always saturated.

It is important to note that admitting flows beyond a certain threshold does not reduce the blocking probability in overload and therefore only deteriorates performance. An optimal choice of admissibility threshold should produce negligible blocking in normal load while maintaining sufficiently high throughput in overload. In the light of the above results, a reasonable compromise for the present system is a threshold between 0.5% and 2% of link capacity. The precise value of the admission threshold is not critical and consequently does not require a highly accurate estimation.

5 Simulation of Integrated Implicit Admission Control

To evaluate admission control under realistic traffic conditions we have performed a number of simulation experiments using ns2, the Network Simulator[1].

5.1 Simulation Model

Network model. We considered the simple dumbbell topology shown in Figure 3. All links have the same fixed delay of 10 ms. Admission control is performed on the 10 Mbit/s bottleneck link. We have verified that the 5 Mbit/s access links do not affect throughput in this configuration. The link buffer has a capacity of 50 packets. Packets of streaming flows have head-of-line priority.

Traffic model. TCP connections are generated by each source node according to a Poisson process. Each connection is used to transfer a stream of 1 Kbyte packets representing a document of a certain size and then terminated. The document size is drawn from the following distribution: 90% of documents are so-called “mice” with size uniformly distributed between 1 and 9 packets; the remainder, deemed “elephants” have size uniformly distributed between 10 and

[1] http://www.isi.edu/nsnam.ns/

400 packets. This choice is made for the sake of simplicity. Performance is largely independent of the size distribution, as noted in Section 4.

Streaming traffic is generated in the form of UDP connections and represents 20% of the offered load. Flow duration is drawn from an exponential distribution with a mean of 60 sec. All UDP packets have a fixed size of 200 bytes. We considered two different traffic models, CBR and on/off. In the CBR model, the UDP connections have a fixed constant rate equal to 15 kbit/s. In the second model, connections are intermittent with the transmission rate in an on-period ten times that of the CBR connections and the probability of being in an on-period equal to 0.1. On- and off-period durations follow an exponential distribution.

5.2 TCP Phantom

The TCP phantom connects the extremities of the bottleneck link. Its goodput, equal to the rate of acknowledged packets, is measured in fixed time intervals of 0.1 s and available bandwidth is estimated by applying exponential smoothing to these measurements.

Figure 4 shows how the blocking probability depends on the admission threshold in underload ($\rho = 0.9$) and overload ($\rho = 1.4$). The figure contrasts the simulation results with the predictions of the theoretical fluid model of Section 4. Figure 5 plots the throughput realized by the phantom connection as a function of the admission threshold and Table 1 compares the throughput realized by the admitted connections depending on their size.

We observe that the throughput achieved by TCP flows depends on their size, whereas it did not in the fluid model. The throughput of mice is severely limited by TCP slow start even when the number of simultaneously admitted flows is small. The throughput of large transfers of more than 100 packets depends less on the slow start limitation. The permanent phantom has the highest throughput of all.

In line with the predictions of the fluid model, we observe that there is no advantage in choosing a threshold smaller than 0.5%. A threshold of 0.5% yields blocking which is negligible in underload and no more than the fluid limit $(\rho - 1)/\rho$ in overload. A threshold lower than 0.5% increases response times

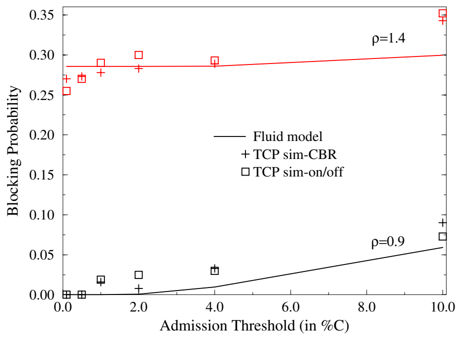


Fig. 4. Blocking probability

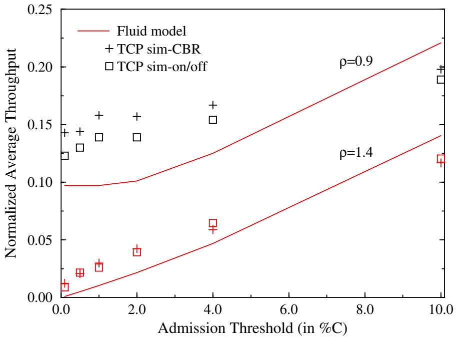


Fig. 5. Normalized phantom throughput

Table 1. Impact of rate threshold on realized throughput (in %), $\rho = 0.9, 1.4$

		$\rho = 0.9$			$\rho = 1.4$		
Threshold		All	> 100	Phantom	All	> 100	Phantom
0.5%	CBR	2.1	6.0	14.4	1.4	2.1	2.7
	on/off	2.0	5.2	13.0	1.3	1.9	2.2
2%	CBR	2.1	6.0	15.7	1.6	3.2	4.2
	on/off	2.1	5.7	13.9	1.6	3.3	3.9
4%	CBR	2.1	6.1	16.7	1.7	3.8	5.9
	on/off	2.1	5.5	15.4	1.7	3.8	6.5

and does not reduce the blocking probability. For a threshold higher than 4%, the theoretical results are somewhat optimistic. Such a threshold tends to be inefficient since the admitted connections are not able to completely saturate the link, as assumed in the fluid model, yielding high blocking and no compensating increase in throughput. The simulations thus confirm that any value between 0.5% and 2% constitutes an acceptable admission threshold.

The results are broadly the same for CBR and on/off streaming flows. Elastic flow throughput is somewhat less for the on/off streaming traffic, probably because TCP cannot adjust the rate sufficiently rapidly. However, the same conclusions apply with respect to the choice of threshold.

5.3 Loss Rate

In this section we evaluate the effectiveness of the second admission control approach based on the measured loss rate. In the simulation, we simply measure the loss rate averaged over all TCP packets in 0.1 second time intervals and apply exponential smoothing.

It proves difficult to precisely calibrate the observed loss rate with a target available bandwidth. Table 2 gives the average throughput realized by all flows and by large flows ($>$ 100 packets), respectively, together with the blocking rates

Table 2. Impact of loss threshold on realized throughput (in %) and on blocking (in %), $\rho = 0.9, 1.4$

Loss threshold		$\rho = 0.9$			$\rho = 1.4$		
		All	>100	Blocking	All	>100	Blocking
1%	CBR	3.3	13.9	13.7	2.8	11.4	40.8
	on/off	3.3	13.5	14.0	2.9	11.3	42.5
5%	CBR	2.9	10.7	1.2	2.0	5.5	28.8
	on/off	2.8	9.9	1.3	2.0	5,0	30.0
10%	CBR	2.8	10.5	0	1.5	2.8	24.9
	on/off	2.7	9.5	0	1.5	2.52	27.1

for different loss thresholds. We notice hardly any difference between the results obtained with the CBR traffic model and those relative to the on/off model.

A threshold smaller than 1% is overly conservative and leads to significant blocking in normal load. For thresholds greater than 5% blocking is relatively stable and roughly equal to the fluid limit. Realized throughput decreases as the threshold increases, notably for large transfers.

6 Conclusion

We have proposed an integrated admission control scheme applying to both streaming flows and elastic flows. This scheme uses implicit flow rejection and measurement-based admissibility conditions avoiding the need for signalling and per-flow resource reservation. It consists in estimating the rate a new elastic flow would acquire and accepting a new (streaming or elastic) flow only if it would not reduce the throughput of ongoing elastic flows below a certain threshold. Giving priority to packets of streaming flows minimizes loss and delay for audio and video applications without penalizing elastic flows whose minimum throughput is guaranteed.

In order to investigate the choice of optimal admission threshold, we have developed an analytical fluid model integrating both classes of traffic under the quasi-stationary assumption. The model was shown using simulation to provide a good approximation under rather general and realistic traffic assumptions. The range of optimal admission threshold values predicted by the model was confirmed by simulations taking into account UDP and TCP dynamics at packet level. We found that any threshold in the range of 0.5% to 2% of the link capacity is appropriate. This range of values coincides with that proposed in [4] where only elastic traffic was offered. It appears that streaming traffic in the considered proportion (up to 20%) has little impact in this respect. One of the conclusions that can be drawn from this study is that the choice of threshold is not critical so that the estimation of available bandwidth does not need to be highly accurate.

In order to estimate the available bandwidth, two algorithms were evaluated and shown to work satisfactorily. One of the algorithms relies on a TCP phantom connection and the second is based on measuring the packet loss rate. The inherent tolerance of elastic flows to rate fluctuations and the fact that streaming traffic is admitted at a relatively low load level partly explain the effectiveness of both algorithms. In our scheme remains a critical issue regarding the feasibility of flow identification on the fly at high speed interfaces. A test bed is currently being set up to evaluate feasibility and performance in a real network environment.

References

1. A. Kumar, M. Hegde, and S.V.R. Anand. NETMASTER : Experiences in Using Nonintrusive TCP Connection Admission Control for Bandwidth Management of an Internet Access Link. *IEEE Communications Magazine*, May 2000.

2. R. Mortier, I. Pratt, C. Clark, and S. Crosby. Implicit Admission Control. *IEEE Journal on Selected Areas in Communications*, December 2000.
3. J.W. Roberts and S. Oueslati-Boulahia. Quality of Service by Flow Aware Networking. *Phil. Trans. Royal Society London*, 2000.
4. S. Ben Fredj, S. Oueslati-Boulahia, and J.W. Roberts. Measurement-based Admission Control for Elastic Traffic. In *ITC 17*, 2001.
5. J.W. Roberts. *Self-similarity in network traffic*, chapter Engineering for Quality of Service. Wiley, 1999.
6. S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J.W. Roberts. Statistical Bandwidth Sharing: A Study of Congestion at Flow Level. In *SIGOMM*, 2001.
7. J.W. Roberts, U. Mocci, and J. Virtamo. *Broadband Network Teletraffic, Final Report of European Action COST 242*. Springer, 1996.
8. T. Bonald, A. Proutière, and J.W. Roberts. Statistical Performance Guarantees for Streaming Flows using Expediated Forwarding. In *INFOCOM*, 2001.
9. L. Massoulié and J.W. Roberts. Bandwidth sharing and admission control for elastic traffic. *Telecommunications Systems*, 15:185–201, 2000.
10. R. Gibbens and P.F. Kelly. Measurement-based Connection Admission Control. In *ITC 15*, volume 2b, pages 879–888. Elsevier, 1997.
11. Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: A Simple and Effective Flow Control Scheme. In *SIGOMM*, 1996.
12. J.W. Cohen. The Multiple Phase Service Network with Genaralized Processor Sharing. *Acta Informatica*, 12:245–285, 1979.

Novel Enhancements to Load Control - A Soft-State, Lightweight Admission Control Protocol

A. Marquetant[1], O. Pop[1], R. Szab□[2], G. Dinny□s[1], and Z. Tur□nyi[2]

[1]Department of Telecommunications and Telematics
Budapest University of Technology and Economics
Stoczek 2, Budapest H-1111, Hungary
{marquet, pop}@ttt-atm.ttt.bme.hu
[2]Ericcson Telecommunications Ltd.
Laborc u. 1., Budapest H-1037, Hungary
{Robert.Szabo, Zoltan.Turanyi}@eth.ericsson.se

Abstract. In this paper we present enhancements to a load control algorithm, which was proposed to perform resource reservations in a differentiated services domain. The basic load control algorithm performs distributed admission control decision, uses only aggregated network states (one per per-hop-behavior □PHB) and soft state approach with periodic refreshes to account for resource usage. It is enhanced by several means to adapt it to recent networking requirements. Our proposals include solution to multi-rate reservation, enhancements in network utilization adapting the protocol to transient periods, and handling of possible quality of service violations. We also demonstrate through prototype implementation and extensive performance analysis to what extent the former objectives are fulfilled by our proposals. We claim that most of the enclosed proposals in this article could be reused in other lightweight resource reservation protocols.

Keywords: admission control, resource reservation/provisioning, differentiated services

1 Introduction

For several years now the networking trends focus on the enhancement of the ever most successful packet network (the Internet) in order to carry multiple traffic types. Even if this unified transport network is founded primarily on datacom principles, it is expected to provide traditional telecommunication services by means of strict quality of service (QoS) requirements. Understanding the evolution more and more packet networks will carry real-time, call-based traffic. This traffic will consist of not only telephony calls, but also other voice, audio or video based traffic such as streaming media or videoconference applications. Nevertheless, the former service integration is unlikely to happen until methods to ensure the required QoS service models, like advanced packet scheduling, buffer management mechanism and call admission control (CAC), are not researched, developed and deployed in the Internet.

As an answer to these requirements the Internet Engineering Task Force (IETF) has proposed two architectures based on the granularity of service differentiation. The first one, in chronological order, was the Integrated Services (IS) [1] approach where the service separation used the finest possible granularity by means of handling each flow individually. Still, this model is generally believed to have bad scaling properties concerning core networks. The supplementary IETF solution, which was named Differentiated Services (DS) [2], is best described by way of class-based service differentiation and lightweight, if any, signaling mechanism to provide QoS. Independently of the former architectures certain traffic management algorithms are indispensable in the network to meet application requirements. Any practical traffic management method is categorized either by its time-scale of operation or by its underlying control capability. In this fashion such algorithms can operate on packet and burst or call and connection scale. In addition, their control authority classifies them to the group of either preventive or reactive algorithms. Packet service disciplines are good examples for packet and burst level algorithms whose task is to provide end-to-end performance bounds via classifying and scheduling the traffic. Among call and connection level preventive traffic control approaches admission control is one of the most widely known ones. Based on the location of admission decision one can differentiate between a distributed and a centralized case. Typical of the distributed class are Resource reSerVation Protocol (RSVP) [8], Boomerang [9] and others described in [10] [11] [12] [13] [14]. Centralized approaches are recently very popular in differentiated services networks where the referred control entity is generally named bandwidth broker (BB) [15]. Another way to categorize admission control algorithms is the information they rely on when making decisions. Generally, this information can be either collected by measurements [17] [18] or by administrating reservations [8] [9] [10] [11] [12] [13] [14].

None of the existing proposals has proved to be general yet simple enough to worth wide deployment. This is because the best way to provide bandwidth, delay or loss guarantees may heavily depend on the actual network and traffic conditions. A solution that is feasible in a campus environment may completely fail in a company intranet, a wireless access network or in a transcontinental backbone. The selection of the proper QoS architecture may depend on factors including user profiles, the available equipment and management budget, the extent to which QoS should be supported, policies and the way the network is managed and built. After nearly a decade of research there is no clear consensus on how to provide QoS, which may indicate that one single solution may never solve all related problems.

This paper is a continuation of an earlier work [16] □ named Load Control lightweight signaling protocol □ that focused on the development of a preemptive traffic management algorithm employing the distributed decision model in the company of either measurement or reservation based loading state collection. The protocol development of the basic Load Control scheme has ever since been continued and an enhanced proposal version recently hit the IETF standardization body under the name of Resource Management in Diffserv (RMD) [20] [21]. The authors of this article are involved in the former standardization, though this paper discusses general, performance related considerations that led to the enhancement version about to be standardized. This way, describing protocol details is out of the scope of this paper, and the reader is further referred to the standardization work at

IETF. Rather, problems and performance issues that raised from earlier standardization are generally investigated.

The rest of the paper is organized as follows: Section 2 revisits related works in the resource management field. After that the original Load Control protocol is introduced with its identified weaknesses (section 3) along with some functional enhancements in order to better adapt to nowadays networking environments (section 4). This is pursued by an extensive prototyping based evaluation of competing proposals in section 5. Finally, in section 6 conclusions are drawn.

2 Related Works

Within the Internet community there is a long history of congestion control research [1] [4] [5]. Most of these efforts addressed bulk data transfer applications and the improvements were built into the Transmission Control Protocol (TCP). Recent results in the field introduce explicit congestion notification (ECN) signals that are sent by routers toward the end systems [6]. These end systems are required to slow down when receiving such signals. While this approach — especially when coupled with the random method suggested for generating these signals [7] — is applicable to data transfer applications, it is less suitable for non-adaptive real-time traffic.

The utility of data transfer degrades gracefully with decreasing available bandwidth. In contrast, non-adaptive traffic significantly suffers from the least insufficiency of available bandwidth. Therefore the preferred way of handling non-adaptive traffic is to block additional requests whenever the network is close to congestion.

There exist several resource provisioning or traffic control algorithms to address the admission control and resource management problem described above. One approach is to use a signaling protocol to communicate the resource needs from end systems to routers, where a corresponding resource reservation state is stored [8] [9] [10]. These approaches are very close to the Integrated Services [1], and require every network entity to maintain per-flow state related information, which spoils the objective of lightweight mechanism hallmarked by the differentiated services architecture.

Another broad class of candidate algorithms don't even require per flow state information handling but rather maintaining aggregated states in network-core devices. These algorithms generally assume soft reservation status in the network and either aim to periodically update it [11] [12] or try to harmonize the actions of routers along the communication path [14] or take an economic approach to handle congestion [13].

All former approaches assumed distributed decision model, on the other hand, by centralizing the resource management and provisioning, core routers are relaxed of maintaining flow related information. Bandwidth Brokering (BB) [15] depicts a fiction where a central device manages a DiffServ domain. Then again this model as well has its disadvantages, namely that there is a single point of failure and a bottleneck if request intensity is too high.

3 The Load Control Protocol

Load control as originally described in [16] targeted the DS architecture by providing edge-to-edge load management functions in a DS domain, or between two RSVP-capable routers, where only the edge devices keep flow states and do per-flow processing. The main purpose of Load Control was to provide a simple and scalable solution to the dynamic resource-provisioning problem by performing two action types: *i)* controlling the admission of arriving request and *ii)* reacting on exceptional events like link failures. The execution of admission control was further divided into two operation types namely *a)* simple marking, where load information is collected through measurements and *b)* unit based reservation, where a bandwidth share or unit is periodically signaled from edge-to-edge to maintain reservation status. The idea was to use two-bit markers in the IP packet headers to collect load information as well as to perform admission control.

Henceforth in this article we focus on the unit based reservation model when discussing performance issues. For this reason we briefly revisit its operation along with its weaknesses.

The unit based approach operates in a distributed fashion; it does not require per flow states but a single aggregated status for each differentiated services code point (DSCP) within the networks□core routers and uses the soft state paradigm to account for and time-out resource usage. The protocol identifies two basic functions namely *i)* the limitation of real-time traffic along any communication path by doing call admission control and *ii)* the maintenance of reservation soft states by periodically updating resource usage along the paths. Note that real-time traffic could further be subdivided into classes according to the differentiated services paradigm, where the DSCPs are used to mark each subclass. The underlying idea of Load Control is to use bit markings in packet headers (e.g., in the DS byte or ECN field), in view of the fact that CAC decision has typically a boolean (yes or no) outcome. Thus, Load Control related information could be piggybacked on existing packet traffic avoiding any unnecessary signaling overhead. Also by placing simple control information into packet headers, most Load Control related operations could be implemented in the fast data path, which may assume hardware solutions. A proposed implementation appeared in [16], where the two basic functions are as follows:

First, let□s assume that there exists an agreement on the resource unit within the administrative domain. Usually it is a bandwidth unit describing effective bandwidth, peak rate or average rate. At this point, whenever a new connection request arrives to the ingress edge router of a domain it injects as many special packets marked as *probe* towards the destination as many resource units the flow requires. Assume that *probe* packets are binary encoded by 0x01. Core routers of the network maintain state variables as follows: *last* and *count* for each DSCP class, and the length of the refresh period denoted by *T*. Whenever a request (*probe*) packet passes a core router it first checks whether *last+1* is within the resource budget (*resource probing*) and if so it admits the flow and increments both the *last* and the *count* variables. Note that admitted *probe* packets are unchanged. On the other hand if the request is rejected it is remarked to 0x11 to signal it. Downstream routers along the path cannot unmark an already rejected *probe* packet and must regard it as ordinary data packets. Egress edge routers echo back the IP header of *probe* (0x?1) packets to inform the corresponding ingress edge about the request□s status. There is another special packet type named *refresh*, which is marked by 0x10. Network ingress edge routers update reservations by sending as many *refresh* packets in each refresh period (*T*), as many units have to be kept reserved. In each refresh interval core routers sum these packets in the state variable *count,* which is then copied to state variable *last* at the end of *T*. Note the requirement of per reservation state maintenance at the edges. In contrast core routers simply count the passing *refresh* packets in each *T* period for all DSCPs. It is the decision and the duty of the network operator to unify the resource unit and the refreshment period network wide.

3.1 Concerns Regarding the Load Control Protocol

Load Control as originally designed was very simple which resulted in certain drawbacks. In this subsection we identify some problems of the protocol for which we propose some alternative solutions in the next section.

3.1.1 Multi-unit Reservations

There is a very large number of applications with diverse resource needs in the current Internet. This diversity also manifests in bandwidth demands, which naturally calls for handling connections with more than a single reservation unit. The former train of thoughts directly led to the consideration of multi-unit reservation requests.

Given that the original idea proposed standard IP header encoding of reservation requests (*probe* packets), the flexibility of signaling multi-unit reservations in this header is limited due to the lack of available unused/un-standardized bits. Consequently one separate reservation request packet has to be sent for each unit of the multi-unit reservations. This can introduce loss in resource utilization when, for instance, more than one edge routers want to reserve units at the same time through the same core router, and the available units are enough for only one of the connections. In this case as the *probe* packets may be overlapped in time, none of the connections will be admitted. Additionally, if request packets are transmitted back-to-back to reduce the formerly mentioned exaggerated race condition then unnecessary burst are added to the network. Moreover, evenly distributed request packets will add to the admission delay since the last request packet must also be collected before making decision. Also the more request packets to transmit the higher their loss probability is. Besides, there is the question of granularity; the smaller the unit is, the finer the resolution and the smaller the overbooking is but the more request packets need to be transmitted too.

Similar arguments could be enumerated for the reservation update case, however there is no race condition since only *refresh* packets of the previous interval affect admission control. Moreover the timing criterion is absent too.

3.1.2 Reservation Transients

Reservation transients appear when transforming from resource probing to reservation update as well as when terminating reservations. The former case can be explained by highlighting that resource requests (e.g. BW[units]; where BW>>1) may arrive at any time within a refresh period. At this point, reservation refresh updates are periodically transmitted in each T/BW (<<T) interval, however resource requests already allocated the necessary resources for the refresh period it arrives in (see Fig. 1, where BW=5; message zero represents 5 *probe* packets for the connection request, and 1,2,□ are reservation *refresh* messages). Note that the sample session is overbooked in its resource request period by 2 units. In general, the sooner the resource request arrives after the beginning of a refresh period and the more bandwidth it requests, the higher the overbooking is.

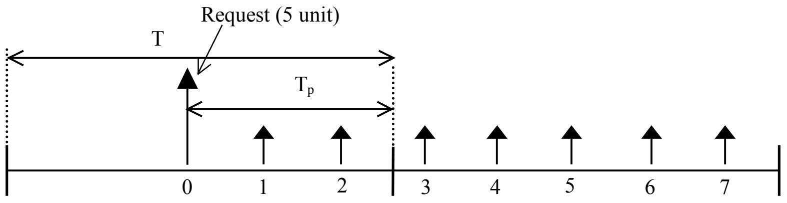


Fig. 1. Transient period at the beginning of a connection

In addition, when a reservation terminates, all its registered updates have to be timed out, which takes at worst two refresh periods (the proof is left to the reader). During this time, which can last for seconds or even minutes, a new connection request may be rejected, although it would be possible to admit. Thus, large refresh periods can significantly reduce network utilization, as we will show at numerical results section. The adjustment of the refreshment period is a trade-off between accuracy and overhead. Generally, the longer the refresh period is the more exact the estimation of the resource usage is and the less processing and signaling overhead is introduced; yet also the more resources are wasted during soft state time-outs and the slower the architecture□s reactions are at topology changes. Protocol overhead is even exaggerated if no piggybacking is used.

Here again, the closer the last update is to the beginning of the refresh period the longer it takes to time out but the less the overbooking is; in contrast the later the last *refresh* arrives in a refresh period the sooner it times out though the more it over allocates.

Altogether, both transients are disadvantageous and therefore must be controlled.

4 Enhanced Load Control

In this section we detail all our proposals to handle the above-described limitations of the original load control algorithm. First the question of multi-unit allocation is investigated with possible solutions proposed. It is then followed by solutions to transient behavior and at last but not least possible QoS violation of the architecture is discussed.

4.1 Multi-unit Reservations

A straightforward extension of the original protocol would be to extend *probe* packets to signal the required number of resource units *explicitly*. In this way one can avoid the multi-unit reservation□s weakness described in section 3.1.1. Another benefit of the multi-unit extension is that connection establishment will only take one round-trip time. However, we point out the fact that the standard IP header can not cope with encoding resource unit demand in *probe* packets. Therefore, an optional header, individual not piggybacked *probe* packet or other solutions are required to do this encoding, which leads to more complex packet handling in core routers. Nevertheless *probe* packet processing is done only once per connection. Thus for multi-unit reservations connection establishment time will still be reduced.

The same technique can be also used for the *refresh* packets. However we argue for leaving these packets unmodified for the following reasons:

4.1.1 Network Jitter

The network jitter that is introduced by routers□queues may drift *refresh* packets in a way that they appear in the same refresh interval, although they are intended for adjacent intervals. In case of explicit multi-unit refreshing, the resulting over/under allocation of resources is equal to at least the number of units refreshed in a single *refresh* packet (see Fig. 2). On the contrary, if using single unit refresh updates piggybacked by ordinary data packets, and marking data packets at every T/BW_{unit}, only a fraction of the *BW* is subject to the effect of network jitter (see Fig. 3).

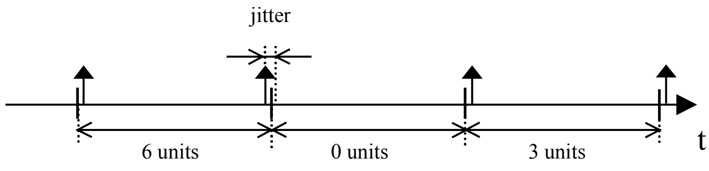


Fig. 2. Effect of network jitter: explicit refresh case

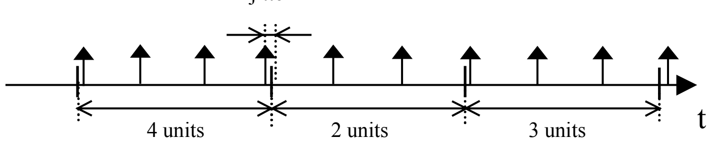


Fig. 3. Effect of network jitter: single-unit refresh case

4.1.2 When Reservation Ends

Consider that one session terminates right after its last *refresh* packet, which arrives at the very beginning of a refresh period (see Fig. 4-left). From the operation of load control, its reservation times out only after two refreshment periods. However, by using the single-unit refresh mechanism the closer the termination time is to the beginning of the refresh period the less unnecessary resource is allocated in the two consecutive periods (see Fig. 4-right). Note, that in the situation where the last *refresh* packet arrives just before the end of a refresh period, both approaches will allocate unnecessary resources for an extra refresh period.

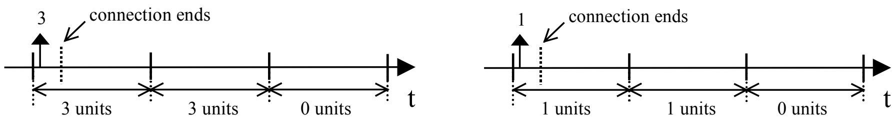


Fig. 4. Connection end (left □explicit, right □single unit)

4.2 Reservation Transients

From the operators□point of view, network utilization plays an important role. Hence we investigated three approaches to avoid the algorithm□s pitfalls in achieving high utilization due to reservation transients. In the following subsections we present a mechanism, which handles transient over-allocation when transforming from resource probing to reservation update, then an explicit release method to overtake the problem of soft state time-outs and finally we detail a sliding window algorithm that refines the resolution of the refresh period.

4.2.1 Enhanced Probe Processing

In order to avoid the transient over-allocations at the beginning of a connection we propose the following mechanism, which we named enhanced probe processing. *Probe* packets are carrying the requested resource units (*BW*) and core routers take admission control action based on their state variable *last*; hence the update of *last* is not modified. However, when updating *count*, one can account for the possible future *refresh* packets that may arrive in the same refresh period. Remember that at the end of each refresh period *count* is copied to *last*. For proper operation one must ensure that successful requests increment *count* as necessary, i.e., at the end of the refresh period at least but close to *BW* increment is added by the request. In our proposal, core routers update their *count* by $\lceil BW(1- T_p/T) \rceil$ unit, where T_p is the time left till the next refresh interval and T is the length of the refresh period (see Fig. 1). Note here, that we do not require any synchronization in the system or any further knowledge of the edges. The only requirement is that the single-unit *refresh* packets are evenly distributed in the refresh interval. Each router can individually calculate the bandwidth portion to increment its resource usage with. Another advantage of this approach is that if a request fails at a later node, it will unnecessarily allocate only the calculated portion in the subsequent refresh period.

4.2.2 Explicit Resource Release

The problem with explicitly releasing resources is that routers cannot easily decide how many refreshes have already arrived for a given connection that must be released still within the current interval. A straightforward solution would be to register the arrival times of each connection□s *refresh* packets, which would introduce per flow states in network core routers. This is obviously against the lightweight paradigm of differentiated services and would overly complicate the algorithm itself.

If one cannot address the problem explicitly then alternative solutions to approximate ideal operation are explored. For the evenly distributed one-unit refresh approach, one can easily estimate the approximate number of *refresh* packets sent in a refresh period until time T_r, when the number of received updates are T_r/T * BW (see Fig. 5, where BW=5; message P represents the release packet, and P-1, P-2, □ are reservation *refresh* messages). Hence, we propose an operation, where state variable *last* is decremented by the released resource amount (*last* $-=BW$), while *count* is only reduced according to the former portion, i.e., *count* $-= \lfloor BW * T_r / \mathrm{T} \rfloor$. As it can be seen, this is analog with the enhanced probe processing mechanism.

Note the conservative approximation of the bandwidth to be released. This is to avoid possible QoS violation that is addressed later.

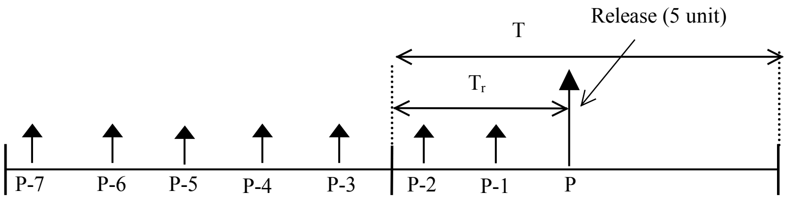


Fig. 5. Explicit release of resources

4.2.3 Sliding Window Algorithm

Another idea to cope with transients at the end of reservations was to split the refresh interval (windows size - *T*) in a fixed number of cells (*N*) (see Fig. 6). Let□s denote the one cell interval by *dT=T/N*. Now, update the original algorithm to maintain *N* x *count* and the *last* state variables. The algorithm is further changed that at time *t*, *last* is updated traditionally, however only *count[i] i* ∍ *[0, ..., N-1]* is incremented, where *i*= $\lfloor t/dT \rfloor$ mod *N*. The former equation means that only the variable *count[i]*, that is assigned to the cell the request or *probe* packet arrives to, is incremented, hence one knows how many reservations and updates arrived in each of the cell intervals. After each cell time (*dT*), the sum of *count[i] i=0...N-1* is copied to *last* and *count[i], i=* $\lfloor t/dT \rfloor$ mod *N* is reset to zero. The former extension just works fine in general, i.e., it is easy to see that the over allocation of resources after the termination of a connection falls back to one refresh period. Also note, that the introduced state variables in the core routers are still constant, hence the algorithm scales independently of the number of flows. However, our proposed algorithm to solve the problem of reservation transients (see sections 4.2.1 and 4.2.2) cannot be easily adopted into this algorithm. Therefore, we propose a minor change to the algorithm in order to make it capable of handling reservation transients.

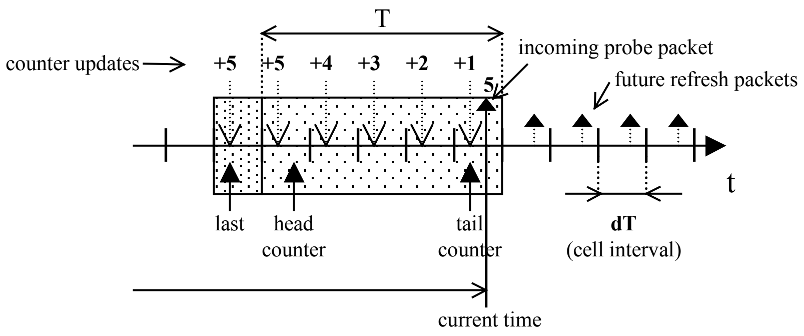


Fig. 6. Sliding window algorithm

Transients

In order to avoid transient over allocations we introduce a fundamental change in the approach we handle sliding windows. We propose that instead of storing the number of *refresh* updates arrived in a certain cell in the *count* variables, the *sum* of refreshes should be maintained in a way that at time *t*, *count[i+n* mod *N], i=* $\lfloor t/dT \rfloor$ mod *N* and *n=[0...N-1]* stores the sum of refreshes back to *N-n* cells, i.e., the counter at the head (*i=* $\lfloor t/dT \rfloor$ mod *N*) stores a value corresponding to the whole refresh period, the next stores a value corresponding to the whole refresh period but one cell, and the tail counter stores a value corresponding to a single cell only. Note, the former states that each *refresh* packet is accounted in each counter. After each cell interval the *last* state variable is updated to the headcounter, which is set to zero afterwards. Now the tricky thing, where resource requests are accounted as follows: assume a request for *BW* at time *t*, which is T_p time to the next cell interval. Now

$$count[i+n \bmod N] = \lceil BW/BW_{unit}\,\{1-(n*dT+T_p)/T\}\rceil,$$
where $i=\lfloor t/dT \rfloor \bmod N$ and $n=[0...N-1]$.

Note that the underlying idea is just the same that we introduced in section 4.2.1, however we update each counter by considering how further *refresh* packets will arrive in their upcoming intervals before they update the *last* counter.

The integration of release messages to the sliding window approach must be handled similarly as presented for the requests; thus the exact algorithm is left for the readers. Furthermore, as we consider conservative decisions in both cases, i.e., at the resource allocation and at the resource release, the probability of QoS violation is reduced (see next section).

4.3 QoS Violation

In the context of load control we call QoS violation the situation when the sum of the admitted connections resource demands exceeds the available resource of at least one core router in the domain. We identified at least one source of this misbehavior as the delay jitter suffered by the load control signaling packets. Note that at enhanced probe packet processing we conservatively over-estimate the bandwidth units to be reserved and at the receipt of an explicit release packet we make an underestimation regarding the number of units to be released. However, if there is any network jitter, both the traditional algorithm and the enhanced ones are subject to temporary QoS violation. We have to rely on the fact that sources are not synchronized and aggregating many flows at core routers yields nearly as many overbookings as underbookings statistically.

5 Performance Analysis

To carry out performance evaluation of our proposed algorithm we have implemented a simple test-bed that can bee seen on Fig. 7. Our source and core router comprises a PC with modified Linux kernels. We generate calls according to Poissonian arrival and exponential holding times. The arrival intensity of the calls was chosen 0.7 calls/second, while the mean holding time was set to 60 seconds. The purpose behind these values was to achieve around 50% blocking probability at scenarios described below. The main performance measures of interest are network utilization and QoS violation degree. The prototype implementation uses ICMP *echo* packet to signal both *probe* and *release* requests, while *refresh* updates are piggybacked by ordinary data packets.

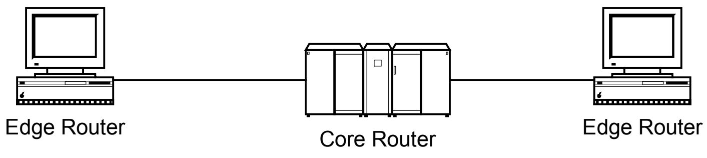


Fig. 7. Load control test-bed scenario

In all of our measurements, where not stated otherwise, we compare the performance of the traditional approach without explicit release and enhanced probe processing with an implementation supporting explicit release and enhanced probe packet handling. We combined both approaches with the sliding window algorithm. Note that in the special case of having one cell and without our enhancements, we get the original Load Control algorithm.

5.1 Network Utilization

In the following charts we present utilization measurements for two scenarios. In the first case we investigated one-unit reservation requests in a system that could allocate 62 bandwidth units. Results are presented with and without explicit release.

These results are followed by a reservation scheme having multi-unit requests scaling from 1 bandwidth unit to 20 bandwidth units evenly distributed. Here the system□s offered traffic increased 10 times the value of the previous case in average, hence the bandwidth allocation threshold was set to 620 units to assure the same level of blocking. Note, that our main objective was not to provide a real traffic scenario but rather to test the performance of the algorithms in extreme situations.

5.1.1 Single Unit Reservations

In Fig. 8 network utilization is shown for different refresh periods without explicit release. One can notice that the utilization is quite low, however the number of sub windows deployed within the refresh interval significantly increases the achieved utilization. Also notice the effect of refresh time on utilization, i.e., the shorter the refresh time is the higher the achieved utilization is because the unnecessarily reserved resources after the termination of a connection are swept off in proportion to the refresh time. This assumption is further verified in Fig. 9, where one can see that by using explicit release and enhanced probe processing the impact of the selection of refresh period length is insignificant.

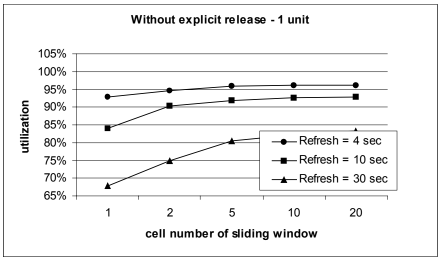


Fig. 8. Network utilization with sliding window, without explicit release and without enhanced probe processing- single unit case

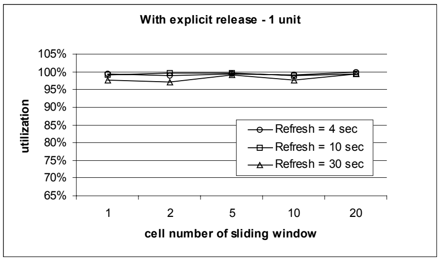


Fig. 9. Network utilization with sliding window and with explicit release and enhanced probe processing- single unit case

5.1.2 Multi-unit Reservations

These measurements show the results in a multi-rate reservation environment (see Fig. 10 and Fig. 11). Note the similar characteristics already detailed in the single rate environment.

Based on these results one could conclude that the sliding window is less effective in the case of multi-unit environment, since the utilization gain is much less with the increasing cell numbers as compared to the single unit case (see Fig. 8 and Fig. 10).

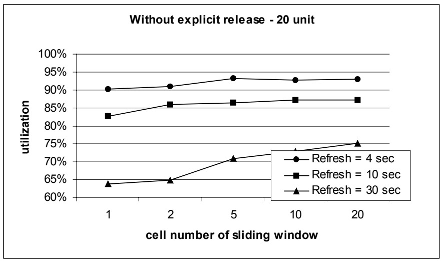


Fig. 10. Network utilization with sliding window, without explicit release and without enhanced probe processing- multi-unit case

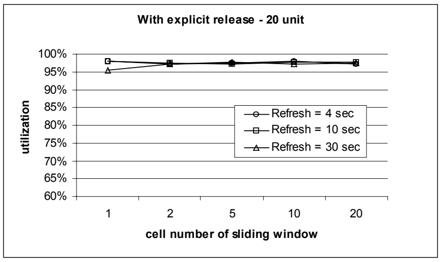


Fig. 11. Network utilization with sliding window, with explicit release and enhanced probe processing- multi-unit case

5.2 QoS Violations

Fig. 12 and Fig. 13 show the QoS violation results measured in the two basic scenarios that were used in the previous measurements. First of all note that there were no QoS violations for the traditional approach, without explicit release and enhanced probe processing. This however is not considered as a pitfall of the extensions but rather the poor utilization of the traditional approach (see Fig. 8 and Fig. 10). Also note that the multi-rate scenario operates at a lower utilization (Fig. 11) compared to the single unit reservation (Fig. 9), hence as shown in Fig. 13 the QoS violation is significantly less than the single unit case shown in Fig. 12.

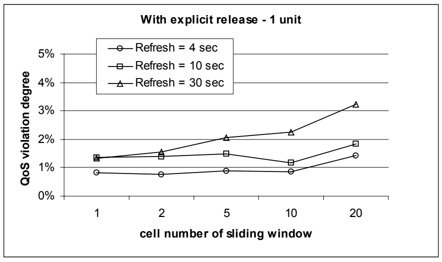


Fig. 12. QoS violation with sliding window with explicit release and enhanced probe processing single unit case

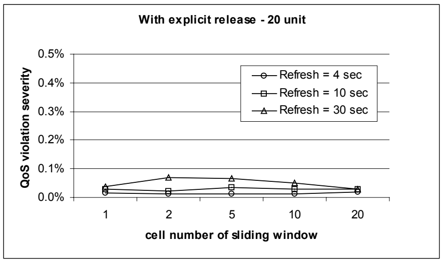


Fig. 13. QoS violation with sliding window, with explicit release and enhanced probe processing multi-unit case

5.3 Comparison of Enhancements

Finally, we investigated the benefits of different enhancement steps we introduced to the traditional load control algorithm. We illustrate our results with a configuration like the one RSVP uses. It can be seen in Fig. 14 how individual enhancements improve network utilization. It is also notable, that these enhancements are required to achieve high utilization regardless of the introduced complexity.

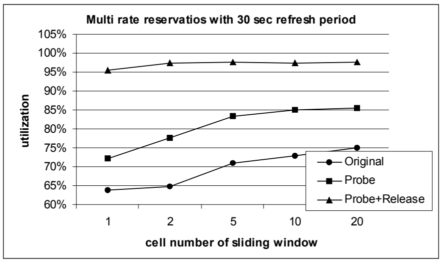


Fig. 14. Comparison of various enhancements applied to load control

6 Conclusions

In this paper we have presented enhancements to Load Control [16], a lightweight resource reservation protocol for DiffServ networks. We have proposed and investigated several algorithms to improve the poor network utilization of the original proposal. Some of these proposals were already incorporated to an enhanced protocol draft recently submitted to IETF [20] [21] while others remain alternatives or are subject to further analysis. Note that the main focus of this work was to carry out protocol improvements in network utilization. Similar efforts were made on other protocol aspects like profiling, fairness etc, however these are left out due to space limitation.

All described mechanisms were implemented and tested in a Linux-based DiffServ network, and then extensive performance evaluation was made to compare them with the basic protocol. The results show that the former objective was reached.

Acknowledgements

This work was carried out in a joint cooperation between the High Speed Networks Laboratory of the Department of Telecommunications and Telematics, Budapest University of Technology and Economics and Ericsson Telecommunications Ltd. Special thanks to Lars Westberg, Georgios Karagiannis, Vlora Rexhepi, Martin Jacobsson and Simon Osthoek for their investments in the load control enhancements.

References

1. R. Braden, D. Clark and S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, IETF RFC-1633, Jun. 1994
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Whang. W. Weiss, “An Architecture for Differentiated Services”, IETF RFC-2475, 1998
3. V. Jacobson, “Congestion Avoidance and Control”, ACM SIGCOMM’88, Palo Alto, 1988
4. S. Floyd, Congestion Control Principles, Internet Engineering Task Force, Jun. 2000.
5. Srisankar Kunniyur and R. Srikant, “End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks”, IEEE INFOCOM’2000, Tel Aviv - Israel, Mar. 2000
6. K. Ramakrishnan and S. Floyd, “A Proposal to add Explicit Congestion Notification (ECN) to IP”, IETF RFC-2481, Jan. 1999
7. Sally Floyd and Van Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397--413, Aug. 1993.
8. B. Braden, B. Ed., et al., “Resource Reservation Protocol (RSVP) - Version 1 Functional Specification”, IETF RFC-2205, Sept. 1997
9. G. Fehér, K. Németh et al., “Boomerang - A Simple Protocol for Resource Reservation in IP Networks”, IEEE Workshop on QoS Support for Real-Time Internet Applications, Vancouver - Canada, Jun. 1999.
10. Ping Pan and Henning Schulzrinne, “YESSIR: a simple reservation mechanism for the Internet”, ACM Computer Communication Review, vol. 29, no. 2, pp. 89--101, Apr. 1999.
11. W. Almesberger, T. Ferrari, J.-Y. Le Boudec, “SRP: a Scalable Resource Reservation Protocol for the Internet”, Computer Communications, vol. 21, no. 14, pp. 1200-1211, Sep. 1998.
12. A. Eriksson, “Resource reservation in a connectionless network”, Proc. of Performance Information and Communication Systems (PICS’98), 1998.
13. R.J. Gibbens, F.P. Kelly, “Resource pricing and the evolution of congestion control”, Automatica 35, 1999
14. Stoica, H. Zhang, “Providing Guaranteed Services Without Per Flow Management”, SIGCOMM’99, Boston, Oct. 1999
15. Internet 2 QBone Bandwidth Broker Advisory Council homepage, http://www.merit.edu/working.groups/i2-qbone-bb
16. L. Westberg., Z. R. Turanyi, D. Partain, “Load Control of Real-Time Traffic”, Internet Draft <draft-westberg-loadcntr-xx.txt>
17. G. Bianchi, A. Capone, C. Petrioli, "Throughput Analysis of End-to-End Measurement-Based Admission Control in IP", IEEE INFOCOM 2000
18. Viktória Elek, Gunnar Karlsson and Robert Rönngren, “Admission Control Based on End-to-End Measurements”, IEEE INFOCOM 2000
19. D. Partain et al. “Resource Reservation Issues in Cellular Access Networks”,IETF’s I-D: draft-partain-wireless-issues-00.txt, April, 2001
20. L. Westberg et al, “Resource Management in Diffserv (RMD) Framework”, IETF’s I-D: draft-westberg-rmd-framework-00.txt, April, 2001
21. L. Westberg et al, “Resource Management in Diffserv On DemAnd (RODA) PHR”, IETF’s I-D: draft-westberg-rmd-od-phr-00.txt, April, 2001

PBAC: Probe-Based Admission Control

Ignacio Más Ivars and Gunnar Karlsson

Department of Microelectronics and Information Technology
Royal Institute of Technology (KTH)
S-16440 Kista, Sweden
{nacho, gk}@it.kth.se

Abstract. Different end-to-end measurement based admission control (MBAC) schemes have recently been proposed to support quality of service for real-time data. All these designs share the idea of decentralizing the admission decision by requiring each end host or gateway to probe the network before sending data. This probing provides certain measurements on the network status that can be used to accept or reject the incoming flow. In this paper, we study a probing procedure to perform an admission decision for a controlled load service (CLS). The admission control offers a reliable upper bound on the packet loss for the new session even with short probing phase durations (e.g. half a second). Our probing mechanism only requires the routers to differentiate between two classes of packets: high priority data and low priority probes. Some simulation results on the performance of the scheme are presented.

1 Introduction

In today's Internet, it is widely accepted that the support for real-time data is far from being satisfactory. The service quality of the best-effort service is not predictable and reliable enough to satisfy the user needs for applications with inelastic data flows. Some of this applications, like streaming audio or video demand quite stringent QoS requirements, like a low delay or jitter, that cannot be guaranteed with the current architecture. For example, with ordinary voice conversations, one way delays over two hundred milliseconds are often annoying. Most multimedia applications are designed to manage losses that occur in a slightly loaded network and to smooth out the jitter under this network condition. Recently, several schemes have been proposed for measurement based admission control (MBAC) that try to assure some quality of service [1][2][3][4][5][6][7][8]. In this paper, we propose a probe-based admission control (PBAC) scheme which provides a reliable upper bound to the packet loss probability a flow will suffer in the network. We build our results on the work previously published by Elek et al. [1]. In the current paper, we add a more detailed description of the PBAC procedure and simulation results for many of the questions left open in that article.

The proposed PBAC scheme fits into a controlled-load service (CLS), which is part of a reduced service-set architecture currently under development in our group [1][9]. This architecture includes three different service classes: a guaranteed service with no packet loss due to congestion and deterministic bounds on the delay, a controlled-load service with an upper bound on packet loss and low delay, and a best-effort service that uses the capacity left by the other two classes (see Fig. 1). Both the guaranteed service and the controlled-load service are allocated fixed parts of the link capacity. The parts taken together should be less than the total capacity to avoid starvation of the best-effort service. Note that best-effort traffic can use both capacity that has not been reserved by the two other service classes, as well as reserved capacity that the classes do not use. This architecture only requires the routers to differentiate among three different classes, and to have three service class queues. The low delay in both guaranteed and controlled-load services is bounded by using small, packet-scale buffers in the routers.

To perform the loss measurement for the controlled-load service we use probe packets that are sent at the peak rate of the new session. We require the queueing system of this service class to be able to differentiate data packets from probe packets, so that probes do not disturb ongoing data sessions. To achieve this, two different approaches are possible. In the first one we have double queues, one with high priority for data and one with low priority for probes. In the second approach, we have just one queue with a discard threshold for the probes. The choice between these two approaches can be left as a decision for the router design. We have tested both to show that the behavior is similar and that they can achieve the same performance. The results are presented in Section 3.2.

The rest of the paper is organized as follows: in Section 2 we describe the probing procedure and discuss the different parameters involved; Section 3 contains the simulation results obtained for our scheme; we give a short summary of related work in Section 4, and offer our conclusions and open issues in Section 5.

2 Procedural Description

The purpose of our probing scheme is to prevent new sessions from degrading the QoS of ongoing sessions below some pre-established level. We are, thus, preventing congestion from occurring, rather than resolving it after it happens.

In Fig. 2 we can see the phases of our session establishment scheme. When a host wishes to set up a new flow, it starts by sending a constant bit rate probe at the maximum rate that the data session will require. The probing time is chosen by the sender from a range of values defined in the service contract. This range forces new flows to probe for a sufficient time to obtain an accurate enough measurement, while prohibits them from performing unnecessary long probes. The probe packet size should be small enough so that we get sufficient number of packets in our probing period to perform the acceptance decision. When the host sends the first probe packet, it includes the peak bit rate and length of the probe, as well as a session sequence number in the data field of the packet. With this information we allow the end host to perform an early rejection, based on the expected number of packets that it should receive not to surpass the target loss probability. The probe also needs to contain a session identifier to allow the end host to distinguish probes for different sessions, since we consider that one sender could open more than one session simultaneously. The IP address in the probes would consequently not be enough to differentiate them. The sender starts a timer, which value should be over the probe length plus the expected round trip time. This timer will go off in case the sender does not receive an acknowledgement to the probe. The timer will also allow the sender to infer that none of the probe packets went through and to give up further attempts to setup a connection.

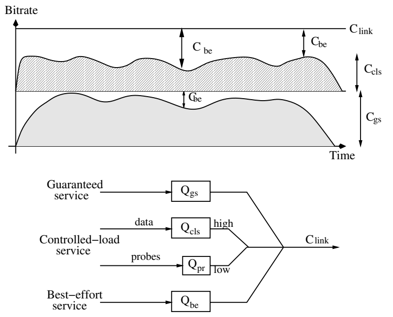


Fig. 1. The separation of the service classes and the corresponding queueing system.

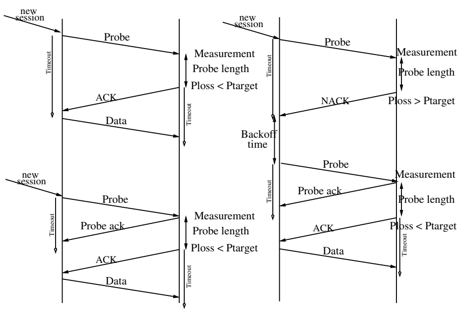


Fig. 2. The probing procedure.

Upon receiving the first probe packet for a session, the end host starts counting the number of received packets and the number of lost packets (by checking the sequence number of the packets it receives). When the probing period finishes and the end host receives the last probe packet, it compares the probe loss measured with the target loss and sends back a packet accepting or rejecting the incoming session. This accept/reject packet is sent back at high priority to minimize the risk of loss. If the decision is positive, the receiver starts a timer with the same value as the timeout in the sender to control the arrival of data packets. If this timer goes off, the host assumes that the acceptance packet has been lost and resends it.

It is important to notice than these two timeouts do not affect the performance of the acceptance decision. They are used for garbage collection in the end hosts so that there are no pending sessions that will not be accepted. The timer values only relate to the time an end host waits in an idle state to be accepted in the system, or to receive data from a source that has probed the path.

Finally, when the sending host receives the acceptance decision, it starts sending data with high priority, or, in the case of a rejection, it backs off for a certain amount of time, before starting to probe again. In subsequent retries, the incoming host can increase the probe time length, up to the maximum level allowed, so that a higher accuracy on the measurement is achieved. There is a maximum number of retries that a host is allowed to perform before having to give up. The back off strategy and the number of retries affect the connection setup time for new sessions and should be carefully tuned to balance the acceptance probability with the expected setup delay.

The acceptance threshold is fixed for the service class and is the same for all sessions. The reason for this is that the QoS experienced by a flow is a function of the load from the flows already present in the system. Considering that this load depends on the highest acceptance threshold among all sessions, by having different thresholds we are degrading all flows to the QoS required by the one with the less stringent requirements. The class definition should also state the maximum data rate allowed to limit the size of the sessions that can be set up. Each data session should not represent more than a small fraction of the service class capacity (in the order of 1%), to ensure that statistical multiplexing works well. The procedure, of course, relies on a common trust between the hosts and the network. The particular means to ensure the proper host behavior remains as a research issue.

3 Performance Evaluation

In this section we want to evaluate different aspects of our probing scheme, like the performance of the two different queueing systems, the probe loss distribution, and our admission decision.

3.1 Simulation Methodology

All the simulations performed have been carried out with the network simulator NS-2[1]. In order to understand the behavior of our probing scheme, we used a simple one link scheme for most of the simulations (see Figure 3). The results for a multilink scenario obtained in [1] show that the highest loaded link dominates the behavior and that the scheme discriminates against flows with longer paths. The setup used in this paper contains a variable number of sources sending data to one router which implements our CLS scheme. The output link connects to another router that also contains our scheme. To this second router we have attach a traffic sink which will measure the different performance values. All the simulations, unless noted otherwise, were performed with the double queue scheme already described, with a queue length of two packets for low priority (probes), and a queue length of eight packets for high priority (data). The difference between the two possible queueing architectures is only quantitative and not qualitative, as will be shown in Section 3.2.

In order to capture the behavior of the probing procedure under different scenarios, we used sources with exponential and pareto distributed on–off times, and traffic sources driven from traces of encoded video from different films[2].

[1] http://www-mash.cs.berkeley.edu/ns/

[2] http://www-tkn.ee.tu-berlin.de/research/trace/trace.html

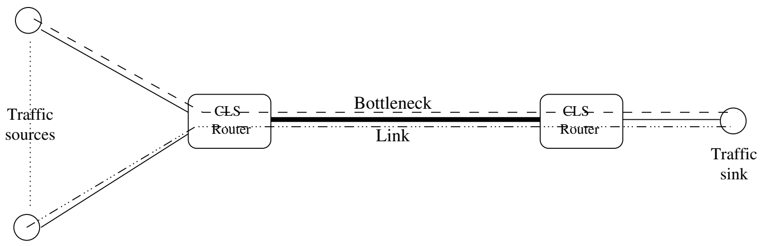


Fig. 3. A simple one link topology network.

These traffic sources were used with different parameter values, like average on and off times and peak rates. The different scenarios used are summarized in Table 1. All the links in the topology were 2 Mb/s for Type1 scenarios (Exp1, Par1, Mixed1 and Video traces) and 45 Mb/s for Type2 scenarios (Exp2, Par2 and Mixed2). The Exp3 source was used for the evaluation of short, and oddly behaving sources (as explained in Sections 3.4 and 3.5). With the Type1 scenarios we have tried to model the behavior of an access network with voice over IP sources, while with the second type of scenarios we model a broadband network with high rate multimedia applications. Both sets contain sources with and without heavy tail distributions (Pareto and Exponential sources).

Table 1. Parameters of the different test sources.

Source	Type	Average On Time	Average Off Time	Peak Rate
Exp1	Exponential	325 ms	650 ms	64 kb/s
Par1	Pareto (β=1.5)	325 ms	650 ms	64 kb/s
Mixed1	Both	325 ms	650 ms	64 kb/s
Exp2	Exponential	20 ms	35.5 ms	5 or 10 Mb/s
Par2	Pareto (β=1.5)	20 ms	35.5 ms	5 or 10 Mb/s
Mixed2	Both	20 ms	35.5 ms	5 or 10 Mb/s
Exp3	Exponential	500 ms	500 ms	256kb/s
Video Traces	-	-	-	-

3.2 Evaluation of the Two Queueing Schemes

In order to understand the effect that the two proposed queueing schemes would have for our admission control, we performed a set of tests with both queues and compared the results. The tests were run by fixing a background load of ongoing sessions on the network and running over 1400 probing sessions for each load, source type and queue type used. The background load was provided by exponential or pareto on–off sources with the parameters listed in Table 1. As we previously mentioned, the buffer sizes of the double queues were two packets for the probe queue and eight packets for the data queue. The threshold queue

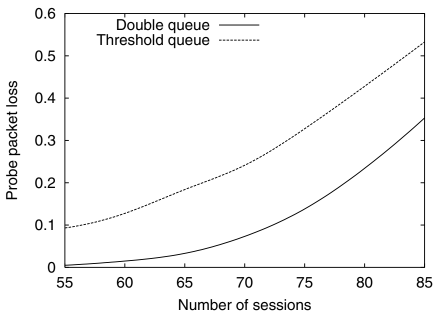


Fig. 4. Probe loss with the two different queue schemes for Exp1 sources.

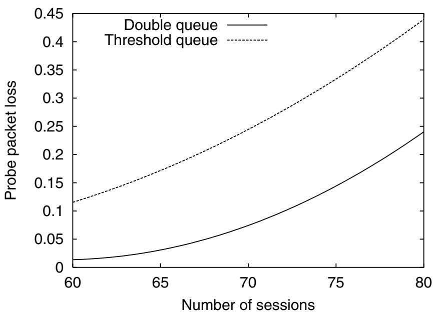


Fig. 5. Probe loss with the two different queue schemes for Par1 sources.

had eight packets of buffer space in total, with a threshold value of two packets. Figures 4 and 5 show the results for exponential and pareto sources, respectively.

From these figures, we notice that there is similar behavior in both probe loss curves, the main difference being the magnitude of the loss that we get for the probing. As expected, the probe loss in the case of the threshold queue is a little bit higher than the one we would achieve for the double queue system, since the two buffer positions below the threshold are shared between high and low priority packets. The loss levels are however always in the same order of magnitude for all load levels we have tested. We conclude that the design decision of choosing one queueing system or the other does not affect the feasibility of our admission control, and both queueing systems could be used together in a network, as long as we take into account the increased loss for the threshold.

We have also tested how much we should increase the discard threshold on the single queue to achieve a similar probe loss than the double queue, in order to provide a suggestion on the values that should be considered for the router buffers. The results obtained for an Exp1 scenario (see Table 1) can be seen in Figure 6. We can see that with a threshold value of four or five packets, depending on the load of the link, we achieve similar probe losses as with the double queues. The other scenarios show similar behaviors, requiring between three to five positions below the threshold to achieve comparable losses as to the other queueing scheme.

3.3 Evaluation of the Admission Decision

To perform the acceptance decision, we define a target loss probability p_{pr} and measure the empirical loss probability p_{me} from the probe. The accuracy of the measurement, as we have previously said, depends on the number of probe packets, so we should use the smallest packet size possible. We have been using a probe packet size of 64 bytes for Type1 scenarios and 100 bytes for Type2 scenarios. This last value has been kept to test previous results obtained with another simulator [1]. These values give us around 125 packets for each probing period in the first set of scenarios and thousands of packets in the second set, if we take one second probing duration in both cases.

For our first test of the admission decision rule we assumed a normal distribution of the probe loss [1]. This assumption allows us to define a confidence level on the measured loss probability, so that a session is acceptable if: $p_{me} + z_R\sqrt{\frac{p_{me}\times(1-p_{me})}{s}} \leq p_{pr}$, given that $p_{pr} \times s > 10$. In this, s is the number of probe packets sent, R is the confidence level we want to have and z_R is the $1-(1-R/2)$-quantile of the normal distribution. The second condition ensures that we have sufficient number of samples for the estimation.

We have tested this assumption by performing many probe sessions and plotting a histogram of the probe loss values that we have obtained. The results can be seen in Fig. 8. In this figure we see the distribution of the probe loss for 750 probe sessions of two seconds each, over a 45 Mb/s link. The packet size is 100 bytes and we have a source of the Exp2 type with 5 Mb/s peak rate (see Table 1).

With the performance evaluation of the admission control, we are trying to show the existence of a general relationship between probe and session loss probabilities. We have performed different simulations and compared the different probe/session loss figures obtained. The results are plotted in Fig. 7. This figure has been obtained for Type1 scenarios with 70 active sources of the different types over 2 Mb/s links. It shows that there is a gap between the expected session loss and the probe loss that depends on the type of sources being used. In general there is one order of magnitude higher probe loss than session loss. A clearer view of the relation between probe and session loss can be seen in Figs. 9 and 10, for the different test scenarios. Figure 9 contains the results obtained by changing the source types among the first set of sources (Type1) over a 2 Mb/s link, while in Fig. 10 we used the second set of sources (Type2) over a 45 Mb/s link. From these figures we see that the actual probe to session loss ratio does not depend heavily on the type of sources, and that there is a nearly linear relationship between the loss types.

3.4 Impact of Short Sessions and Thrashing Probe Behavior

In order to have a high degree of trust in our probe-based admission control, we have tried to evaluate the effect of very short sessions. The reasons are twofold: first, we want to show that with our probing duration (from 0.5 to 2 seconds), we obtain a reliable upper bound on the packet loss probability, even for sessions in the length of the probe period; second, we want to address the issue of thrashing in the case of many simultaneous probes [10]. To make this test we use a scenario in which we have Exp3 sources. The link capacity is 2 Mb/s, and there are five, ten or fifteen sources that try to setup sessions randomly. We have given these sessions a lifetime of 5, 10, 20, 30, 60, 120 or 240 seconds, and have measured the average data and probe loads on the link for 5000 seconds of simulated time. For each simulation, the offered load is fixed and defined as the session arrival rate times the session lifetime. Figures 11, 12 and 13 show the results with a probe

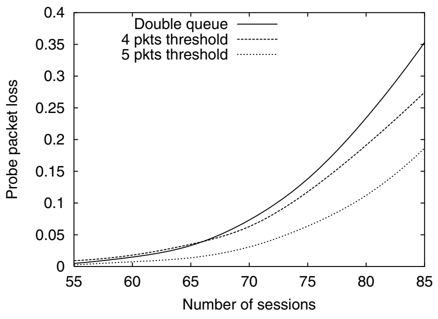


Fig. 6. Probe losses for both queueing schemes with 4 and 5 packets as discard threshold.

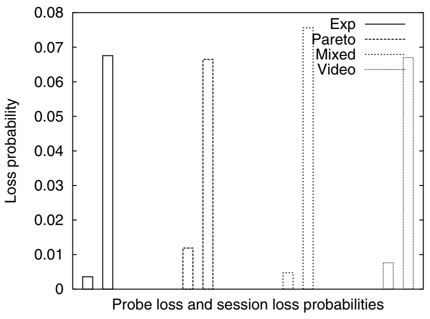


Fig. 7. Probe and session loss probabilities for different scenarios.

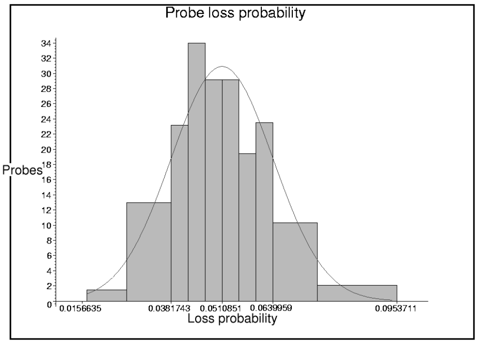


Fig. 8. Probe loss histogram for a Type2 scenario.

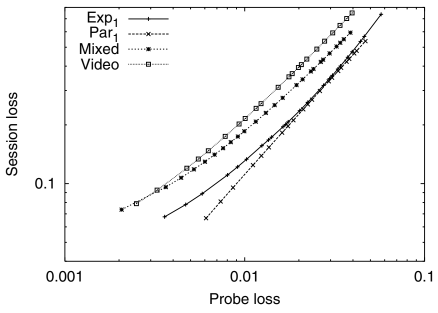


Fig. 9. Sessions versus probe packet loss for the different source types in the Type1 scenarios.

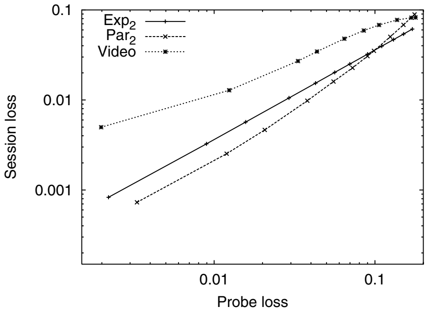


Fig. 10. Sessions versus probe packet loss for the different source types in the Type2 scenarios.

queue of only two packets. From these figures, we see that as we are increasing the session arrival rate, because of the shorter lengths of our sessions, we have a much higher probe load on the system. This increased probe load leads to a higher blocking probability, because of simultaneous probes contending, and thus, to a lower link utilization. If we consider Figures 12 and 13, in which the offered load to the system is high, the utilization only reaches 80% for sessions over 50 seconds in length. For shorter sessions, the increased amount of session arrivals per second (and probe load) is basically too high for new sessions to enter successfully. The existence of short sessions can considerably decrease the utilization by increasing the blocking probability, but it will never increase the loss probability of the ongoing sessions!

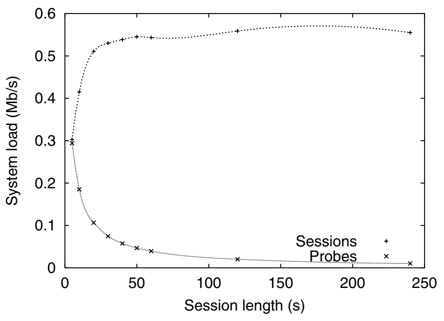


Fig. 11. Accepted sessions load and probe load in a 2 Mb/s link with 5 sources of 256kb/s peak rate.

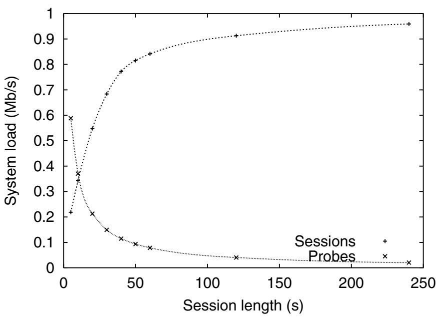


Fig. 12. Accepted sessions load and probe load in a 2 Mb/s link with 10 sources of 256kb/s peak rate.

There is one important point to note when considering these simulation results: they deal with odd behavior in the packet arrival rate of the flows. We have setup the simulation in a way to test the possible thrashing effect. In a real network, reducing the session length does not necessarily increase the call arrival rate, as these are independent variables. We have also used a simple back-off strategy for blocked calls, based on a random waiting time. This time is obtained from a uniform distribution over an interval equal to the probe length and it is used for three subsequent probes. If none of these three probes succeed, then the source waits for another random time uniformly distributed over the sessions length. This particular back-off strategy increases the probe arrival rate. However, it could happen that no matter the length of the ongoing sessions, we receive a burst of session establishment attempts in a particular moment of time. If this is the case, we could possibly incur this thrashing that would not allow any new session to enter in the system. This would be a temporary problem, as the sources would finally abandon their attempts, and the probe load would decrease. The solution is to define a maximum number of allowed retries of session establishment, as we have previously described. Also, consider that in a real system we are expecting to have a high degree of multiplexing, with session peak rates being in the order of 1% of the link capacity, at most, so that the increased probe load for each new session would be fairly small.

There is another consideration to make. For a real router, we should have a probe queue length of at least one packet per input port. We have made a simulation with 10 sessions and have increased the probe queue up to ten packets, to compare the results with Fig. 12. Just by increasing the probe queue length, we have achieved a much better performance in the lower end of the curve, as seen in Fig. 14. It is clear that by having at least one buffer position per session, we substantially decrease the possible thrashing in our system.

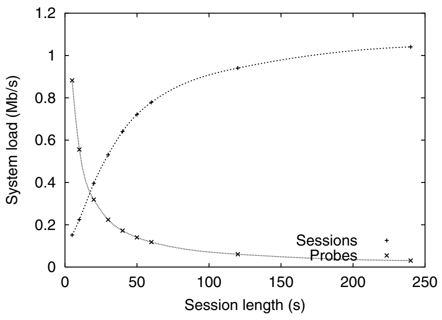


Fig. 13. Accepted sessions load and probe load in a 2 Mb/s link with 15 sources of 256 kb/s peak rate.

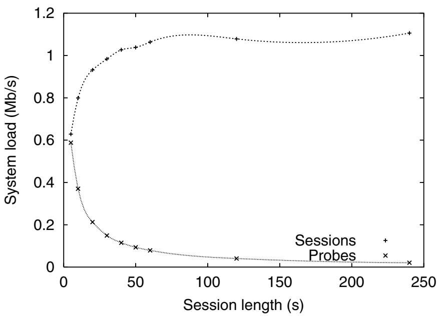


Fig. 14. Accepted sessions load and probe load in a 2Mb/s link with 10 sessions and 10 probe queue buffers.

To have a feeling about what the situation would be in a real environment, we performed one last set of simulations fixing the session arrival rate. The results show what we were expecting. The load on the system is more or less constant even for short session lengths. Figure 15 shows the utilization curves for the same parameters as previous figures, when we fix the call arrival rate to achieve a utilization of 1.4, 1.15 or 0.95 Mb/s.

3.5 Impact of Session Behavior

One of the most important things to consider in the whole admission control scheme is how to enforce a proper behavior of the sessions in the CLS service. There is one situation in which our admission scheme would have problems to accurately predict the data loss, and it is the case in which we have a certain number of sessions that transmit nothing at all for periods of time longer than our probe length. Then, it could happen that these sessions keep silent while a probe is taking place. The probe loss measurement would not take into account these silent sessions, which are still ongoing and might load the network. This situation could lead to a data loss higher than the expected maximum value. However, we have to consider the probability of this particular situation happening. With sessions being in the order of 1% of the link capacity, the fact

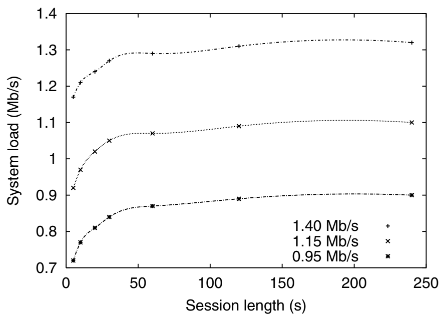


Fig. 15. Accepted sessions load for different load targets in a 2 Mb/s link

that a small number of them are silent during the probing would make almost no difference to the probe loss measurement. The multiplexing of independent sessions makes it highly unlikely for a large number of them to keep silent at exactly the same time. Nevertheless, in order to have a feeling for the real problem, a number of tests have been performed both on Type1 scenarios and with the Exp3 sources (see Table 1). The Type1 scenarios give us 60 to 80 sources attempting to establish sessions, while the Exp3 source creates a scenario where one session has a stronger impact on the network load. First, we compared the measurements of the probe loss and the subsequent session loss with the ones obtained in Section 3.3 (see Fig. 9). We created from one to ten oddly behaving sessions that remained silent for 10 seconds and then continued transmitting. The occurrence of the silence periods of these special sessions were randomly and independently chosen during the session length. As expected, the results showed no significant difference over the ones obtained previously.

Regarding the Exp3 source, the results show an increased load on the link, around 120 kb/s on average, which confirms the fact that we are having one more session ongoing in the system than we were expecting. The effect of this increased load depends heavily on the type of session. For this particular example we found a session data loss slightly higher than the one we were achieving with no badly behaving sessions. We were manually configuring the simulator so that many probes occurred during the silent periods. If we add some randomization to the silences, then on average over a long simulation, the results are similar to the ones with no ill-behaving sources, due to the fact that most of the probes occur when the odd session is active. To sum up, the performance of our system under heavy stress (many simultaneous probes, ill-behaving sessions) is reassuring. All the results have been obtained by carefully configuring the simulation for exhibiting a particularly bad behavior and, still, the system was able to perform relatively well. In general, as the situation worsens, the admission control tends to be conservative, allowing less ongoing sessions, but never failing to keep the data packet loss above the target, except when there are many big sessions (compared to the link capacity) keeping silent together. However, such a situation is avoided by restricting the sessions' peak rates.

4 Related Work

The PBAC scheme was first proposed by Karlsson in [2] and is inspired by a proposal from EPFL, called Scalable Reservation Protocol (SPR) [3]. Our proposed controlled-load service based on PBAC would fit well in the diff-serv [11] framework. The difference with our scheme is that PBAC is developed for on-demand session setup, which is not supported by diff-serv. The design principle of our architecture is to provide light-weight session set-up protocols. We want to place most of the functions, like measurements, outside the network or at its edges, to improve the scalability.

Similar schemes have been proposed in the literature, like the ones of Borgonovo et al. [4] for CBR traffic, Kelly et al. [8], based on a general marking function; and more recently, Bianchi et al. [6], with a scheme based on packet delays, and Mortier et al. [7], studying connection admission control at TCP level. There has also been a recent comparison of different proposals by Breslau et al. [10]. This article shows that the performances of the different admission control algorithms are quite similar. Thus, our conclusion is that simplicity is the most important design goal.

5 Conclusions – Open Issues

We have presented an admission control method based on probing that provides a well-defined upper limit on the packet loss probability for a flow. We have offered simulation results that show a clear relationship between the probe packet loss and the expected session loss. To summarize, we offer an admission control scheme that is simple and offers reliable performance under different network conditions. There are, nevertheless, open issues to address and our research continues. The main problems we want to study are:

– The complete semantic definition of our Controlled Load Service has to be described. There are still questions about the effect on the overall performance of the back-off strategy and the number of setup attempts.
– We would like to address the question of mobility. We think that the fact of not having any reservation state kept in the routers should facilitate the provision of CLS also for mobile nodes.
– We want to implement the scheme in the architecture defined by Karlsson and Orava [9]. We know that the results given by simulations are a good approximation of the reality, but that no simulation source can capture the behavior of real traffic traces.

References

1. V. Elek, G. Karlsson, and R. Rönngren, “Admission control based on end-to-end measurements,” in *Proc. of IEEE INFOCOM 2000* [12], pp. 623–630.

2. G. Karlsson, "Providing quality for internet video services," in *Proc. of CNIT/IEEE ITWoDC 98*, (Ischia, Italy), pp. 133–146, September 1998.
3. T. Ferrari, W. Almesberger, and J.-Y. Le Boudec, "SRP: a scalable resource reservation protocol for the internet," in *Proc. of IWQoS 98*, (Napa, CA), pp. 107–116, May 1998.
4. F. Borgonovo, L. Fratta, A. Capone, M. Marchese, and C. Petrioli, "End-to-end QoS provisioning mechanism for differentiated services," Internet Draft, IETF, July 1998.
5. R. J. Gibbens and F. P. Kelly, "Measurement-based connection admission control," in *Proc. of the 15th International Teletraffic Congress* (V. Ramaswami and P. E. Wirth, eds.), Teletraffic Science and Engineering, (Washington D.C., USA), pp. 879–888, Elsevier Science B.V., June 1997.
6. G. Bianchi, A. Capone, and C. Petrioli, "Throughput analysis of end–to–end measurement-based admission control in IP," in *Proc. of IEEE INFOCOM 2000* [12], pp. 1461–1470.
7. R. Mortier, I. Pratt, C. Clark, and S. Crosby, "Implicit admission control," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2629–2639, 2000.
8. F. P. Kelly, P. B. Key, and S. Zachary, "Distributed admission control," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2617–2628, 2000.
9. G. Karlsson and F. Orava, "The DIY approach to QoS," in *Proc. of IWQoS 99*, (London, UK), pp. 6–8, May 1999.
10. L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint admission control: Architectural issues and performance," in *Proc. of ACM SIGCOMM 2000*, vol. 30, (Stockholm, Sweden), pp. 57–69, August/September 2000.
11. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, IETF, December 1998.
12. *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, (Tel Aviv, Israel), March 2000.

QoS Routing: Average Complexity and Hopcount in m Dimensions

F.A. Kuipers and P. Van Mieghem

Delft University of Technology, Information Technology and Systems
P.O Box 5031, 2600 GA Delft, The Netherlands,
F.A.Kuipers@its.tudelft.nl, P.VanMieghem@its.tudelft.nl

Abstract. QoS routing is expected to be an essential building block of a future, efficient and scalable QoS-aware network architecture. We present SAMCRA, an exact QoS routing algorithm that guarantees to find a feasible path if such a path exists. The complexity of SAMCRA is analyzed. Because SAMCRA is an exact algorithm, most findings can be applied to QoS routing in general.
The second part of this paper discusses how routing with multiple independent constraints affects the hopcount distribution. Both the complexity as the hopcount analysis indicate that for a special class of networks, QoS routing exhibits features similar to single-parameter routing.

1 Introduction: Constrained-Based Routing

Delivering end-to-end Quality of Service (QoS) is widely believed to become an essential asset for the future Internet. Much research has been done (and is continuing) on this topic, which resulted in the proposal of several QoS architectural frameworks including IntServ/RSVP, DiffServ, MPLS and traffic engineering through constrained-based routing. Each of these proposals has some advantages over the others. A future QoS-aware network architecture for the Internet will therefore likely comprise of a combination of several architectural frameworks in order to provide end-to-end QoS in an efficient, stable and scalable manner. In this paper we argue that constrained-based routing is an essential QoS building block for providing efficient QoS solutions. Some examples motivate this statement. In the IntServ/RSVP framework, RSVP tries to reserve a (best-effort) path and then provides QoS through appropriate scheduling, queueing, dropping, etc.. However, because the constraints are ignored while setting up/reserving the path, the reserved path is possibly not the best choice, or even worse, a path may not be found at all (flow blocked), while there is a feasible path available. The same phenomenon can be observed in DiffServ. In DiffServ the local packet handling mechanisms (scheduling, etc.) provide the differentiated service, after which the packets are forwarded along a best-effort path. MPLS is a forwarding scheme where packets are given a label corresponding to a certain MPLS path over which they are forwarded. The combination of MPLS with constrained-based routing to set up a MPLS path seems an intuitively straightforward solution. In conclusion, a future QoS-aware network architecture will benefit from combining constrained-based routing with appropriate packet handling in order to provide efficient QoS solutions.

In section 2 we will present SAMCRA [17], a Self-Adaptive Multiple Constraints Routing Algorithm and analyze its complexity, which is verified through simulations. The second part (section 3) of this paper analyses the hopcount of the shortest path in multiple dimensions. This section is an extension to previous work on the hopcount in the Internet [16]. We conclude this paper with conclusions and speculations in section 4.

2 SAMCRA: A Self-Adaptive Multiple Constraints Routing Algorithm

SAMCRA is the successor of TAMCRA, a Tunable Accuracy Multiple Constraints Routing Algorithm [8], [7]. As opposed to TAMCRA, SAMCRA guarantees to find a path within the constraints, provided such a path exists. Furthermore, SAMCRA only allocates queue-space (= memory) when truly needed, whereas in TAMCRA the allocated queue-space is predefined. The major performance criterion for SAMCRA, namely the running-time/complexity, will be discussed in paragraph 2.2. Similar to TAMCRA, SAMCRA is based on three fundamental concepts: (1) a non-linear measure for the path length, (2) the k-shortest path approach [5] and (3) the principle of non-dominated paths [12]. Before we clarify these three concepts we will first introduce the notations used throughout this paper.

A network topology is denoted by $G(N, E)$, where N is the set of nodes and E is the set of links. With a slight abuse of notation we will also denote by N and E respectively the number of nodes and the number of links. A network supporting QoS consists of link weight vectors with m non-negative QoS measures ($w_i(e)$, $i = 1, ..., m$, $e \in E$) as components. The QoS measure of a path can either be additive in which case it is the sum of the QoS measures along the path (such as delay, jitter, the logarithm of packet loss, cost of a link, etc.) or it can be the minimum(maximum) of the QoS measures along the path (typically, available bandwidth and policy flags). Min(max) QoS measures are treated by omitting all links (and possibly disconnected nodes) which do not satisfy the requested min(max) QoS constraints. We call this topology filtering. Additive QoS measures cause more difficulties: *the multiple constrained path (MCP) problem, defined as finding a path subject to more than one additive constraint (L_i), is known to be NP-complete* [10], [20] and hence considered as intractable for large networks.

We continue by explaining SAMCRA's three basic concepts:

1. Motivated by the geometry of the constraints surface in m-dimensional space, we defined the length of a path P as follows [8]:

$$l(P) = \max_{1 \le i \le m} \left(\frac{w_i(P)}{L_i} \right) \tag{1}$$

where $w_i(P) = \sum\limits_{e \in P} w_i(e)$.

The definition of the path length has to be non-linear in order to guarantee that a retrieved path is within the constraints, i.e. $l(P) \leq 1$. A solution to the MCP problem is a path whose weights are all within the constraints. SAMCRA can also be applied to solve multiple constrained optimization problems, e.g. delay-constrained least-cost routing. Depending on the specifics of a constrained optimization problem, SAMCRA can be used with different length functions, provided they obey the criteria for length in vector algebra. Example length functions are given in [17]. In [11] and [13] TAMCRA-based algorithms with specific length functions are proposed and evaluated. By using length function (1), all QoS measures are considered as equally important. An important corollary of a non-linear path length as (1) is that *the subsections of shortest paths in multiple dimensions are not necessarily shortest paths.* This suggests to consider in the computation more paths than only the shortest one, leading us naturally to the k-shortest path approach.

2. The k-shortest path algorithm [5] is essentially Dijkstra's algorithm that does not stop when the destination is reached, but continues until the destination has been reached k times. This concept is applied to the intermediate nodes i on the path from source node s to destination node d, where we keep track of multiple sub-paths from s to i. Not all sub-paths are stored, but an efficient distinction based on non-dominance is made.

3. The third concept in SAMCRA is that of dominance. A (sub)-path Q is dominated by a (sub)-path P if $w_i(P) \leq w_i(Q)$ for $i = 1, .., m$, with an inequality for at least one i. SAMCRA only considers non-dominated (sub)-paths. This property allows us to efficiently reduce our search-space (all paths between the source and destination) without compromising the solution.

2.1 SAMCRA Meta-code

SAMCRA(G, s, d, L)

G: graph, s: source node, d: destination node, L: constraints

```
1. counter = 0 for all nodes
2. endvalue = 1.0
3. path(s[1]) = NULL and length(s[1]) = 0
4. put s[1] in queue
5. while(queue ≠empty)
6.   extract_min(queue) -> u[i]
7.   u[i] = marked grey
8.   if(u = d)
9.     stop
10.   else
11.     for each v ∈ adjacency_list(u)
12.       if(v ≠previous node of u[i])
13.         PATH = path(u[i]) + (u,v)
14.         LENGTH = length(PATH)
```

```
15.          check all non-black paths at v and PATH
      for dominancy & endvalue $\rightarrow$ mark obsolete paths black
16.          if(LENGTH $\leq$ endvalue and PATH non-dominated)
17.            if(no black paths)
18.              counter(v) = counter(v)+1
19.              j = counter(v)
20.              path(v[j]) = PATH
21.              length(v[j]) = LENGTH
22.              put v[j] in queue
23.            else
24.              replace a black path with PATH
25.            if(v = d and LENGTH < endvalue)
26.              endvalue = LENGTH
```

For a detailed explanation of this meta-code, we refer to [17].

2.2 Complexity of SAMCRA

If N and E are the number of nodes and of links respectively in the graph $G(N, E)$, the queue in SAMCRA can never contain more than kN path lengths, where k denotes the number of feasible (i.e. within the constraints) non-dominated paths that are stored in the queue of a node. Because SAMCRA self-adaptively allocates the queue-size at each node, k may differ per node. When using a Fibonacci (or relaxed) heap to structure the queues [6], selecting the minimum path length among kN different path lengths takes at most a calculation time of the order of $log(kN)$. As each node can at most be selected k times from the queue, the **extract_min** function (explained in [6]) in line 6 of SAMCRA's meta-code takes $O(kNlog(kN))$ at most. The for-loop starting on line 11 is invoked at most k times from each side of each link in the graph. Calculating the length takes $O(m)$ when there are m metrics in the graph while verifying path dominance takes $O(km)$ at most. Adding or replacing a path length in the queue takes $O(1)$. Adding the contributions yields a worst-case complexity with $k = k_{\max}$ of

$$O(kN \log(kN) + k^2 mE) \tag{2}$$

where $k_{\max}$ is an upper-bound on the number of non-dominated paths in $G(N, E)$. A precise expression for k is difficult to find. However knowledge about k is crucial to the complexity of SAMCRA, because a large k could make the algorithm useless. As an upper-bound for k, we will use $k_{\max} = \lfloor e(N-2)! \rfloor$, which is an upper-bound on the total number of paths between a source and destination in $G(N, E)$ [18]. If the constraints/metrics have a finite granularity, another upper-bound applies [8]:

$$k_{\max} = \frac{\prod_{i=1}^{m} L_i}{\max_j (L_j)} \tag{3}$$

where the constraints L_i are expressed as an integer number of a basic metric unit. For instance, if the finest granularity is 1 msec, then the constraint value is expressed in an integer number times 1 msec.

Clearly, for a single constraint ($m = 1$ and $k = 1$), the complexity (2) reduces to that of Dijkstra's algorithm. For multiple metrics the worst-case complexity of SAMCRA is NP-complete.

We will now discuss the complexity of SAMCRA/MCP in more depth. First we will illustrate by an example that an exponential-time algorithm may be better in practice than a polynomial time algorithm: exponential $O(1,001^N)$ versus polynomial $O(N^{1000})$. For any reasonable network size (up to $N = 1,6 \cdot 10^7$) the exponential time algorithm outperforms the polynomial time algorithm. Although this example is exaggerated, there exist some (pseudo)-polynomial ε-approximation algorithms whose running-time is too large for practical purposes and which are therefore only of theoretical interest. Secondly, although there exist many problems that are NP-complete, their average-case complexity might not be intractable, meaning that such an algorithm could have a good performance in practice. The average level of "intractability" differs per NP-complete problem. The theory of average-case complexity was first advocated by Levin [14]. We will now give a calculation that suggests that *the average and even amortized*[1] *complexity of SAMCRA is polynomial in time* for fixed m and all weights w_i independent random variables.

Lemma 1: The expected number of non-dominated vectors in a set of T i.i.d. vectors in m dimensions is upper bounded by $(\ln T)^{m-1}$.

A proof of Lemma 1 can be found by adopting a similar approach as presented in [3] or [2]. Moreover, the results are the same. To gain some insight into the number of non-dominated paths in a graph, we will assume that the path-vectors are i.i.d. vectors[2]. When we apply lemma 1 to the complexity of SAMCRA, we see that in the worst-case SAMCRA examines $T = \lfloor e(N-2)! \rfloor$ paths, leading us to an expected number of non-dominated paths between the source and destination in the worst-case of

$$(\ln T)^{m-1} = (\ln(\lfloor e(N-2)! \rfloor))^{m-1} \leq (1 + (N-2)\ln(N-2))^{m-1}$$

which is polynomial in N. Hence, the amortized complexity of SAMCRA is (2) with $k = (\ln T)^{m-1} = (\ln(\lfloor e(N-2)! \rfloor))^{m-1}$, which is polynomial in time for fixed m.

In the limit $m \to \infty$ and for w_j independent random variables, all paths in $G(N, E)$ are non-dominated, which leads to the following lemma (proved in [17]).

[1] Amortized analysis differs from average-case analysis in that probability is not involved; an amortized analysis guarantees the average performance of each operation in the worst-case [6].

[2] In reality the m weights of the path-vectors will not be i.i.d..

Lemma 2: If the m components of the link weight vectors are independent random variables and the constraints L_j are such that $0 \leq \frac{w_j}{L_j} \leq 1$, then any path with K hops has precisely a length (as defined by (1)) equal to K in the limit $m \to \infty$.

This means that for $m \to \infty$ it suffices to calculate the minimum hop path, irrespective of the weight structure of the m independent components. Since the minimum hop problem is an instance of a single metric shortest path problem, it has polynomial complexity.

Summarizing we can say that if the link weights are independent random variables, there are two properties reducing the search-space (the number of paths to be examined) of SAMCRA. For m small the concept of non-dominance is very powerful, resulting in the presence of only a small number of non-dominated paths between two points in a graph. At the other extreme, for m large, the values L_j of the constraints cause the largest search-space reduction, because only a few paths between the source and destination lie within the constraints. Even if the constraints are chosen infinitely large, SAMCRA may lower them in the course of the computation (by means of `endvalue`, line 25/26 meta-code) without affecting the solution.

The two properties complement each other, resulting in an overall good average performance of SAMCRA. The simulation results of the next paragraph indicate that the average complexity of SAMCRA is $O(N \log N + mE)$, i.e. (2) with $E[k] \approx 1$, for the class $G_p(N)$ of random graphs [4], with independent uniformly distributed link weights.

2.3 Simulation Results on Complexity

The simulations were performed on a large number (minimum of 10^5) of random graphs of the type $G_p(N)$ [4], where p is the expected link-density ($p = 0.2$) and N the number of nodes. The m link weights are independent uniformly distributed random variables.

We will start by presenting the simulation results for $m = 2$. Figure 1 gives the minimum queue-size (k_{min}) needed to find a feasible path. If TAMCRA had used that particular k_{min} under the same conditions, it would have found the same shortest feasible path as SAMCRA did, but if a smaller value had been used TAMCRA would not have found the shortest path. Figure 2 gives the statistics corresponding to the data in Figure 1.

Figures 1 and 2 show that an (exponential) increase of N does not result in a significant increase in k_{min}. The expectation $E[k]$ remains close to 1 and hardly increases with N. If we extrapolate these results to $N \to \infty$, figures 1 and 2 suggest that for the class of random graphs $G_p(N)$ with 2 independent uniformly distributed link weights, the average complexity of SAMCRA is approximately $O(N \log N + 2E)$.

Figures 3-6 show a similar behavior ($E[k] \approx 1$). The only difference is that the worst-case values ($\max[k]$) have slightly increased with m, as a result of the increased expected number of non-dominated paths. However, since $E[k]$ stays

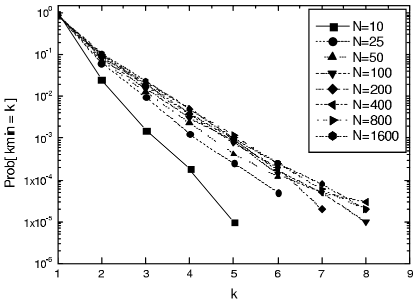


Fig. 1. P.d.f. of $k_{\min}$, m=2

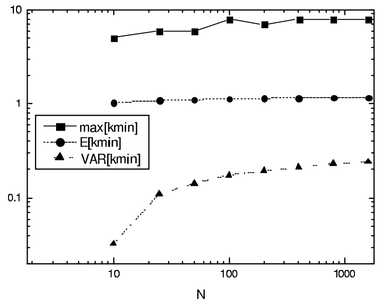


Fig. 2. $k_{\min}$-statistics, m=2

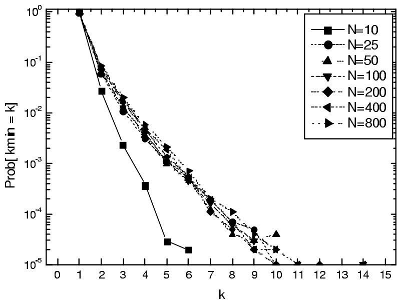


Fig. 3. P.d.f. of $k_{\min}$, m=4

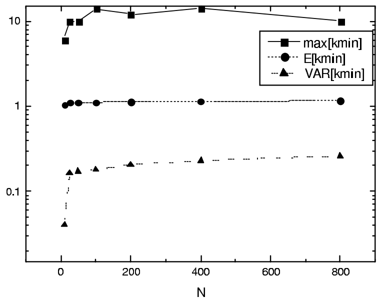


Fig. 4. $k_{\min}$-statistics, m=4

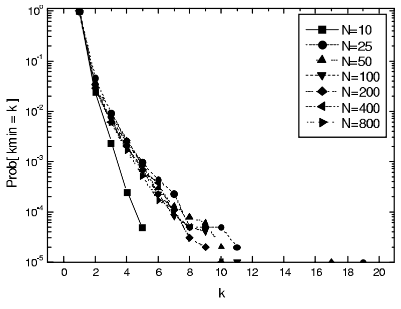


Fig. 5. P.d.f. of $k_{\min}$, m=8

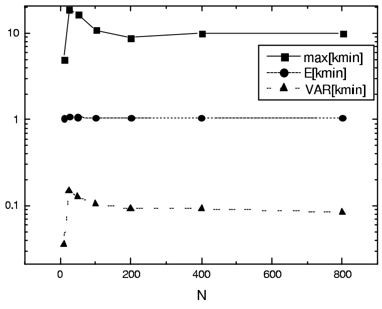


Fig. 6. $k_{\min}$-statistics, m=8

close to one, the simulation results suggest that the two search-space-reducing concepts, dominance and constraint values, are so strong that the average complexity of SAMCRA is not significantly influenced by the number of metrics m. The behavior of $k_{\min}$ as a function of the number of constraints m is illustrated in Figure 7.

In the previous paragraph we indicated that there are two concepts reducing the search-space of SAMCRA. For small m the dominant factor is the non-

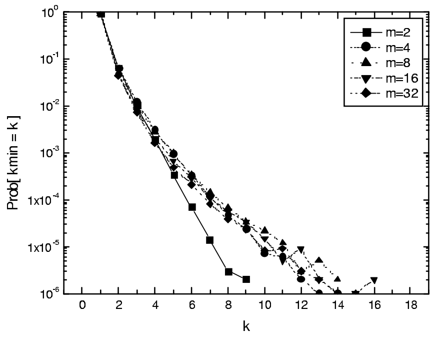


Fig. 7. P.d.f. of $k_{\min}$, N=20

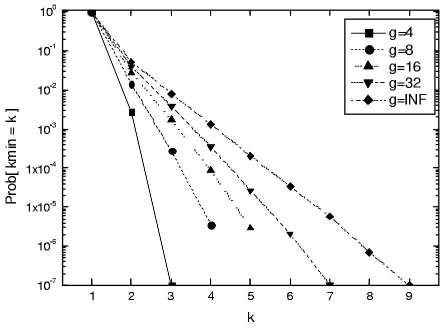


Fig. 8. $k_{\min}$ for different granularity

dominance property, whereas for $m \to \infty$ the constraint values are more dominant. Because these properties are most effective in a certain range of m's, we expect the worst-case behavior to occur with an m that is neither small nor large. Figure 7 shows the k-distribution for different values of m. The best performance is achieved with $m = 2$ and the worst performance is for m around 8. However, as figures 5 and 6 illustrate, $E[k]$ for $m = 8$ is still approximately 1, leading us to believe that for the class of random graphs $G_p(N)$ with independent uniformly distributed weights, the average complexity of SAMCRA is approximately $O(N \log N + mE)$ for every m.

Our last simulation concerns the granularity of the constraints/metrics. When the granularity is finite, an upper-bound, in terms of the constraints, on the number of non-dominated paths can be found (3). The finer the granularity, the larger this upper-bound. Figure 8 confirms this behavior (for $N = 20$ and $p = 0.2$). In practice the constraints/metrics have a finite granularity, which according to Figure 8 will positively influence the running-time of SAMCRA.

Because SAMCRA solves the MCP problem exact, and since the simulations suggest that SAMCRA's average complexity is polynomial for $G_p(N)$ with independent uniformly distributed link weights, the MCP problem for that class of graphs seems, on average, solvable in polynomial time. We have seen that $E[k] \approx 1$, what indicates that routing, for the sizes N considered, in multiple dimensions is analogous to routing in a single dimension ($m = 1$).

The effect that correlation between the link weights has on the complexity of SAMCRA is topic of further study.

3 The Expected Hopcount $E[h_N]$ for the Random Graph $G_p(N)$

As above, we consider the random graph $G_p(N)$ where each link is specified by a weight vector with m independent components possessing the same distribution function

$$F_w(x) = \Pr[w \leq x] = x^{\alpha} 1_{[0,1]}(x) + 1_{(1,\infty)}(x), \quad \alpha > 0 \tag{4}$$

For this network topology, the expected hopcount $E[h_N]$ or the average number of traversed routers along a path between two arbitrarily chosen nodes in the network will be computed. The behavior of the expected hopcount $E[h_N]$ in multiple dimension QoS routing will be related to the single metric case ($m = 1$). That case $m = 1$ has been treated previously in [16], where it has been shown, under quite general assumptions, that

$$E[h_N] \sim \frac{\ln N}{\alpha}$$

$$var[h_N] \sim \frac{\ln N}{\alpha^2}$$

Lemma 2 shows that for $m \to \infty$ in the class of $G_p(N)$ with independent uniformly distributed link weight components, the shortest path is the one with minimal hopcount. Thus the derivation for a single weight metric in [16] for $G_p(N)$ with all link weights 1 is also valid for $m \to \infty$. The first order (asymptotic) calculus as presented in [16] will be extended to $m \geq 2$ for large N. In that paper, the estimate has been proved,

$$\Pr[h_N = k, w_N \leq z] \simeq N^{k-1} p^k F_w^{k*}(z),$$

where the distribution function $F_w^{k*}(z)$ is the probability that a sum of k independent random variables each with d.f. F_w is at most z and is given by the k-fold convolution

$$F_w^{k*}(z) = \int_0^z F_w^{(k-1)*}(z-y) f_w(y)\, dy, \quad k \geq 2,$$

and where $F_w^{1*} = F_w$. By induction it follows from (4), that for $z \downarrow 0$,

$$F_w^{k*}(z) \sim \frac{z^{\alpha k} (\alpha \Gamma(\alpha))^k}{\Gamma(\alpha k + 1)}.$$

In multiple (m) dimensions, SAMCRA's definition of the path length (1) requires the maximum link weight of the individual components $w_N(\gamma) = \max_{i=1,\ldots,m} [w_i(\gamma)]$ along some path γ. Since we have assumed that the individual links weight components are i.i.d random variables, and hence $\Pr[w_N \leq z] = (\Pr[w_i \leq z])^m$, this implies for m-dimensions that

$$F_w^{k*}(z) \sim \left[\frac{z^{\alpha k} (\alpha \Gamma(\alpha))^k}{\Gamma(\alpha k + 1)} \right]^m$$

such that

$$\Pr[h_N = k, w_N \leq z] \simeq N^{k-1} p^k \left[\frac{z^{\alpha k} (\alpha \Gamma(\alpha))^k}{\Gamma(\alpha k + 1)} \right]^m$$

We will further confine to the case $\alpha = 1$, i.e. each link weight component is uniformly distributed over $[0, 1]$.

$$\Pr[h_N = k, w_N \leq z] \simeq \frac{1}{N} \frac{(Npz^m)^k}{(k!)^m} \tag{5}$$

For a typical value of z, the probabilities in (5) should sum to 1,

$$1 = \frac{1}{N} \sum_{k=1}^{N-1} \frac{(Npz^m)^k}{(k!)^m}$$

At last, for a typical value of z, $\Pr[w_N \le z]$ is close to unity resulting in

$$\Pr[h_N = k, w_N \le z] \simeq \Pr[h_N = k]$$

Let us denote with $y = Npz^m$,

$$S_m(y) = \sum_{k=0}^{N-1} \frac{y^k}{(k!)^m} \tag{6}$$

subjected to

$$N + 1 = S_m(y) \tag{7}$$

Hence, the typical value y of the end-to-end link weight that obeys (7) is independent on the link density p for large N. Also the average hopcount and the variance can be written in function of $S_m(y)$ as

$$E[h_N] = \frac{y}{N} S_m^{'}(y) \tag{8}$$

$$var[h_N] = \frac{1}{N}\left[y^2 S_m^{''}(y) + y S_m^{'}(y) - \frac{y^2}{N}\left(S_m^{'}(y)\right)^2\right] \tag{9}$$

We will first compute good order approximations for $E[h_N]$ in the general case and only $var[h_N]$ and the ratio $\alpha = \frac{E[h_N]}{var[h_N]}$ in case $m = 2$. Let us further concentrate on

$$V_m(y) = \sum_{k=0}^{\infty} \frac{y^k}{(k!)^m} \tag{10}$$

Clearly, $V_m(y) = \lim_{N\to\infty} S_m(y)$. In the appendix A it is shown in (21) that

$$V_m(y) = A_m(y) \exp\left[m y^{1/m}\right] \tag{11}$$

with

$$A_m(y) = \frac{(2\pi)^{\frac{1-m}{2}}}{\sqrt{m}} y^{-\frac{1}{2}\left(1-\frac{1}{m}\right)} \tag{12}$$

After taking the logarithmic derivative of relation (11), we obtain

$$V_m'(y) = V_m(y)\left[\frac{A_m'(y)}{A_m(y)} + y^{\frac{1}{m}-1}\right]$$

In view of (7), y is a solution of $V_m(y) \sim N$, such that the average (8) becomes

$$E[h_N] \sim \frac{y}{N} V_m^{'}(y) \sim \frac{V_m(y)}{N}\left[y\frac{A_m'(y)}{A_m(y)} + y^{\frac{1}{m}}\right]$$

or

$$E\left[h_N\right] \sim y^{\frac{1}{m}} + y\frac{A_m'(y)}{A_m(y)} \tag{13}$$

Using (12) in (13), we arrive at

$$E\left[h_N\right] \sim y^{\frac{1}{m}} - \frac{1}{2}\left(1 - \frac{1}{m}\right) \tag{14}$$

where y is a solution of $V_m(y) \sim N$. Equivalently, y is a solution of

$$r(y) = \ln\left[\frac{(2\pi)^{\frac{1-m}{2}}}{\sqrt{m}}\right] - \frac{1}{2}\left(1-\frac{1}{m}\right)\ln y + my^{1/m} - \ln N = 0$$

As initial value in Newton-Raphson's method, for large N, we start with $y_0 = \left(\frac{\ln N}{m}\right)^m$. The next iteration is $y = y_0 + h$, where $h = -\frac{r(y_0)}{r'(y_0)}$. Since

$$r'(y) = \frac{1}{y}\left[-\frac{1}{2}\left(1-\frac{1}{m}\right) + y^{1/m}\right]$$

we have, with

$$Q = \frac{1}{2}\ln m - \frac{1}{2}(m-1)\ln\left(\frac{m}{2\pi}\right),$$

that

$$\begin{aligned} h &= -\left(\frac{\ln N}{m}\right)^m \frac{-Q - \frac{1}{2}(m-1)\ln(\ln N)}{\left[-\frac{1}{2}\left(1-\frac{1}{m}\right) + \frac{\ln N}{m}\right]} \\ &\sim \left(\frac{\ln N}{m}\right)^{m-1}\left[\frac{1}{2}(m-1)\ln(\ln N) + Q + O\left(\frac{\ln(\ln N)}{\ln N}\right)\right] \end{aligned}$$

and

$$y \sim \left(\frac{\ln N}{m}\right)^m + \left(\frac{\ln N}{m}\right)^{m-1}\left[\frac{1}{2}(m-1)\ln(\ln N) + Q\right] + O\left(\ln(\ln N)\ln^{m-2}N\right)$$

From this asymptotic expression for y, we compute

$$y^{\frac{1}{m}} \sim \frac{\ln N}{m} + \frac{1}{2}\left(1-\frac{1}{m}\right)\ln(\ln N) + \frac{Q}{m} + O\left(\frac{\ln(\ln N)}{\ln N}\right)$$

Finally, the average hopcount follows from (14) as

$$\begin{aligned} E\left[h_N\right] \sim \frac{\ln N}{m} + \frac{1}{2}\left(1-\frac{1}{m}\right)\ln(\ln N) + \frac{\ln m}{2m} - \frac{1}{2}\left(1-\frac{1}{m}\right)\left(\ln\left(\frac{m}{2\pi}\right)+1\right) \\ +O\left(\frac{\ln(\ln N)}{\ln N}\right) \end{aligned} \tag{15}$$

This formula indicates that, to a first order, $m = \alpha$. The simulations (Figures 9, 10, 11) show that, for higher values of m, the expectation of the hopcount tends slower to the asymptotic $E\left[h_N\right]$-regime given by (15).

In order to compute the variance, higher order terms in (21) are needed. Although higher order terms can be computed, we confine ourselves to the case $m = 2$, for which $V_2(y) = \sum_{k=0}^{\infty} \frac{y^k}{(k!)^2} = I_0(2\sqrt{y})$ where $I_0(z)$ denotes the modified Bessel function of order zero [1, sec. 9.6]. The variance of the hopcount from (9) with $S_m^{"}(y) = \frac{d^2 I_0(2\sqrt{y})}{dy^2} = \frac{I_0(2\sqrt{y})}{y} - \frac{I_1(2\sqrt{y})}{y\sqrt{y}}$

$$\begin{aligned} var\left[h_{N,2}\right] &\sim \frac{y}{N} I_0(2\sqrt{y}) - \frac{\sqrt{y}}{N} I_1(2\sqrt{y}) + E\left[h_{N,2}\right] - \left(E\left[h_{N,2}\right]\right)^2 \\ &\sim y - \left(E\left[h_{N,2}\right]\right)^2 \end{aligned}$$

At this point, we must take the difference between $I_0(x)$ and $I_1(x)$ into account otherwise we end up with $var\left[h_N\right] \sim 0$. For large x,

$$I_0(x) \sim \frac{e^x}{\sqrt{2\pi x}} \left(1 + \frac{1}{8x} + O\left(x^{-2}\right)\right)$$

and

$$I_1(x) \sim \frac{e^x}{\sqrt{2\pi x}} \left(1 - \frac{3}{8x} + O\left(x^{-2}\right)\right)$$

such that

$$I_1(x) \sim I_0(x) \left(1 - \frac{1}{2x} + O\left(x^{-2}\right)\right)$$

$$\begin{aligned} E\left[h_{N,2}\right] &\sim \frac{y}{N} I_1(2\sqrt{y}) \frac{1}{\sqrt{y}} \sim \frac{I_0(2\sqrt{y})}{N} \left(\sqrt{y} - \frac{1}{4} + O\left(y^{-1}\right)\right) \\ &\sim \frac{\ln(N)}{2} + \frac{\ln\left(\ln(N)\right)}{4} - \frac{1}{4} + O\left(\frac{1}{\ln(N)}\right) \end{aligned} \tag{16}$$

Thus,

$$var\left[h_{N,2}\right] \sim y - \left(\sqrt{y} - \frac{1}{4}\right)^2 = \frac{\sqrt{y}}{2} - \frac{1}{16} + O\left(\frac{1}{\sqrt{y}}\right) \tag{17}$$

and

$$\alpha = \frac{E\left[h_{N,2}\right]}{var\left[h_{N,2}\right]} \sim 2 - \frac{1}{4\sqrt{y}} + O\left(y^{-1}\right) \sim 2 - \frac{\sqrt{2}}{4\sqrt{\ln N}} + O\left(\frac{1}{\ln(N)}\right) \tag{18}$$

This corresponds well with the simulations shown in Figure 9. In addition, the average and variance of the hopcount for $m = 2$ dimensions scales with N in a similar fashion as the same quantities in $G_p(N)$ with a single link weight, but polynomially distributed with $F_w\left[w \leq x\right] = x^2$.

In summary, the asymptotic analysis reveals that, for random graphs of the class $G_p(N)$ with uniformly (or equivalent exponentially) distributed, independent link weight components, the hopcount in m dimensions behaves similarly as

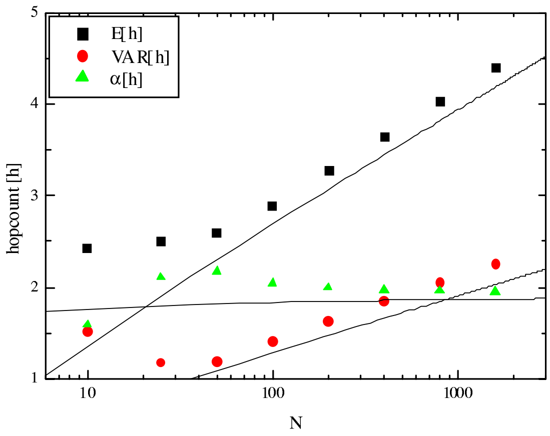


Fig. 9. The average $E\left[h_{N,2}\right]$, the variance $var\left[h_{N,2}\right]$ and the ratio $\alpha = \frac{E\left[h_{N,2}\right]}{var\left[h_{N,2}\right]}$ of the shortest path found by SAMCRA, as a function of the size of the random graph N with two link metrics ($m = 2$). The full lines are the theoretical asymptotics

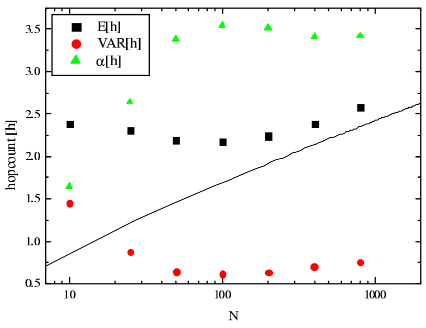


Fig. 10. Hopcount statistics for m=4

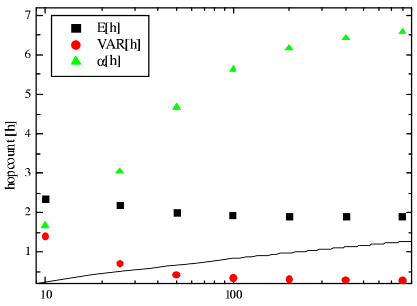


Fig. 11. Hopcount statistics for m=8

in the random graph $G_p(N)$ in $m = 1$ dimension with polynomially distributed link weights specified via (4) where the polynomial degree α is precisely equal to the multiple dimension m. This result, independent of the simulations of the complexity of SAMCRA, suggests a transformation of shortest path properties in multiple dimensions to the single parameter routing case, especially when the link weight components are independent. As argued in [16], the dependence of the hopcount on a particular topology is less sensitive than on the link weight structure, which this analysis supports.

4 Conclusions

Since constrained-based routing is an essential building block for a future QoS-aware network architecture, we have proposed a multiple constraints, exact routing algorithm called SAMCRA. Although the worst-case complexity is NP-complete (which is inherent to the fact that the multiple constraints problem is NP-complete), a large amount of simulations on random graphs with independent link weight components seem to suggest that the average-case complexity is polynomial. For that considered class, the MCP problem thus seems tractable.

The second part of this paper was devoted the study of the hopcount in multiple dimensions as in QoS–aware networks. For random graphs of the class $G_p(N)$ with uniformly (or equivalent exponentially) distributed, independent link weight components, a general formula for the expected hopcount in m dimensions has been derived and only extended to the variance $var\,[h_N]$ as well in $m = 2$ dimensions, in order to compute the variance and the ratio of the expected hopcount and its variance. To first order, with the network size $N >> m$ large enough, the expected hopcount behaves asymptotically similar as the expected hopcount in $m = 1$ dimension with a polynomial distribution function $(x^\alpha 1_{[0,1]}(x) + 1_{(1,\infty)}(x))$ and polynomial degree $\alpha = m$.

Both the complexity analysis as the hopcount computation suggests that for a special class of networks, namely random graphs of the class $G_p(N)$ with uniformly (or equivalent exponentially) distributed and independent link weight components, the QoS routing problem exhibits features similar to the one dimensional (single parameter) case. The complexity analysis suggested this correspondence for small N, whereas the asymptotic analysis for the hopcount revealed the connection for $N \to \infty$. Hence, the question arises whether the QoS routing problem in particular classes of graphs may possess a polynomial, rather than non-polynomial *worst* case complexity. And, further, what is the influence of the correlation structure between the link weight components because simulations suggest that independence of these link weight components seems to destroy NP-completeness. Moreover, we notice that the proof presented in [20] strongly relies on the choice of the link weights. At last, if our claims about the NP-completeness would be correct, how large is then the class of networks that really lead to an NP-complete behavior of the MCP problem? In view of the large amount of simulations performed over several years by now, it seems that this last class fortunately must be small, which suggests that, in practice, the QoS-routing problems may turn out to be feasible.

References

1. Abramowitz, M. and I.A. Stegun (eds.), *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*, Dover Pubns, June 1974, ISBN: 0486612724.
2. Barndorff-Nielsen, O. and M. Sobel, *On the distribution of the number of admissible points in a vector random sample*, Theory of Probability and its Applications, vol. 11, no. 2, 1966.
3. Bentley, J.L., H.T. Kung, M. Schkolnick and C.D. Thompson., *On the Average Number of Maxima in a Set of Vectors and Applications*, Journal of ACM, vol. 25, no. 4, pp. 536-543, 1978.
4. Bollobas, B., *Random Graphs*, Academic Press, London, 1985.

5. Chong, E.I., S. Maddila, S. Morley, *On Finding Single-Source Single-Destination k Shortest Paths*, J. Computing and Information, 1995, special issue ICCI'95, pp. 40-47.
6. Cormen, T.H., C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, 2000.
7. De Neve, H. and P. Van Mieghem, *A multiple quality of service routing algorithm for PNNI*, IEEE ATM workshop, Fairfax, May 26-29, 1998, pp. 324-328.
8. De Neve, H. and P. Van Mieghem, *TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm*, Computer Communications, 2000, vol. 23, pp. 667-679.
9. Edwards, H. M., *Riemann's Zeta Function*, Academic Press, New York, 1974.
10. Garey, M.R. and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.
11. Guo, L. and I. Matta, *Search space reduction in QoS routing*, Proc. of the 19th III Int. Conference on Distributed Computing Systems, III, May 1999, pp. 142-149.
12. Henig, M.I., *The shortest path problem with two objective functions*, European J. of Operational Research, 1985, vol. 25, pp. 281-291.
13. Korkmaz, K. and M. Krunz, *Multi-Constrained Optimal Path Selection*, IEEE INFOCOM 2001.
14. Levin, L.A., *Average Case Complete Problems*, SIAM J. Comput., 1986, 15(1):285-286.
15. Titchmarsh, E. C., *The Theory of Functions*, Oxford University Press, 1964.
16. Van Mieghem, P., G. Hooghiemstra and R. van der Hofstad, *A Scaling Law for the Hopcount in Internet*, Delft University of Technology, report 2000125, 2000, http://wwwtvs.et.tudelft.nl/people/piet/telconference.html.
17. Van Mieghem, P., H. De Neve and F.A. Kuipers, *Hop-by-Hop Quality of Service Routing*, to appear in Computer Networks, 2001.
18. Van Mieghem, P., *Paths in the simple Random Graph and the Waxman Graph*, to appear in Probability in the Engineering and Informational Sciences (PEIS), 2001.
19. Van Mieghem, P., *The Asymptotic Behaviour of Queueing Systems: Large Deviations Theory and Dominant Pole Approximation*, Queueing Systems 23, 27-55, 1996.
20. Wang, Z. and J. Crowcroft, 1996, *QoS Routing for supporting Multimedia Applications*, IEEE JSAC, 14(7): 1188-1234, September 1996.

A Asymptotic Formula for $V_m(y)$

We will apply the Euler-Maclaurin formula

$$\sum_{n=a}^{b} f(n) = \frac{1}{2}\left(f(a)+f(b)\right) + \int_a^b f(t)\,dt + \sum_{n=1}^{N}(-1)^{n-1}\frac{B_{2n}}{(2n)!}\left[f^{(2n-1)}(b) - f^{(2n-1)}(a)\right] + R_N \qquad (19)$$

where B_n are the Bernoulli numbers [1, sec. 23] and

$$R_N = \frac{1}{(2N+1)!}\int_a^b B_{2N+1}(u)\,f^{(2N+1)}(u)\,du$$

Comparison with (10) indicates that $a = 0$, $b = \infty$ and $f(x) = \frac{y^x}{(\Gamma(x+1))^m}$. The derivatives at b all vanish and the derivatives at a follow from the Taylor series of $f(x)$ around $x = 0$,

$$\frac{e^{x \ln y}}{(\Gamma(x+1))^m} = \sum_{k=0}^{\infty} f_k x^k$$

Thus,

$$V_m(x) = \frac{1}{2} + \int_0^{\infty} \frac{y^x dx}{(\Gamma(x+1))^m} - \sum_{n=1}^{N} (-1)^{n-1} \frac{B_{2n}}{(2n)} f_{2n-1} + R_N$$

Let us first concentrate on the integral

$$\begin{aligned} I(y) &= \int_0^{\infty} \frac{y^x dx}{(\Gamma(x+1))^m} \\ &= \int_0^{\infty} \exp\left[x \ln y - m \ln \Gamma(x+1)\right] dx \end{aligned}$$

We will approximate $I(y)$ by the method of the steepest descent for which an exact expansion was published earlier in [19]. Here, we confine ourselves to the leading order term,

$$\int_{-\infty}^{\infty} e^{x[f_1(z_0)\, z - f(z)]}\, dz = \frac{e^{-x[f_0(z_0) - z_0 f_1(z_0)]}}{\sqrt{x f_2(z_0)}} \left(\sqrt{\pi} + O\left(x^{-1}\right)\right) \tag{20}$$

where $f_k(z_0)$ are the Taylor coefficients of $f(z)$ around $z = z_0$. The fastest convergence is expected if z_0 is the maximum of f. Applied to $I(y)$, the maximum of $f(x)$ is the same as the maximum of $g(x) = -x \ln y + m \ln \Gamma(x+1)$. Since $g'(x) = -\ln y + m\psi(x+1)$ where the digamma function equals $\psi(x+1) = \frac{\Gamma'(x+1)}{\Gamma(x+1)}$, we find that the maximum x_0 obeys

$$\psi(x_0 + 1) = \frac{\ln y}{m}$$

or for large y using the asymptotic expansion for the digamma function [1, 6.3.18],

$$\ln(x_0 + 1) - \frac{1}{2(x_0 + 1)} \sim \frac{\ln y}{m}$$

from which $x_0 \sim y^{1/m}$ and

$$\begin{aligned} g(x_0) &\sim -y^{1/m} \ln y + m \ln \Gamma(y^{1/m} + 1) \\ &\sim -m y^{1/m} + \frac{1}{2} \ln y + \frac{m}{2} \ln 2\pi \end{aligned}$$

The higher order derivatives at x_0 of $g(x)$ are polygamma functions $\psi^{(n)}$ [1, 6.4],

$$g^{(n)}(x_0) = m \psi^{(n-1)}(x_0 + 1)$$

The second derivative $g^{''}(x_0) = m\psi'(x_0+1) \sim \frac{m}{x_0+1} + O\left(x_0^{-2}\right)$. Application of (20) yields

$$I(y) \sim \frac{e^{x_0 \ln y - m \ln \Gamma(x_0+1)}}{\sqrt{\frac{m}{2}\psi^{(1)}(x_0+1)}}\sqrt{\pi}$$

$$\sim \frac{e^{my^{1/m} - \frac{1}{2}\ln y - \frac{m}{2}\ln 2\pi}}{\sqrt{\frac{m}{2y^{1/m}}}}\sqrt{\pi}$$

or

$$V_m(y) \sim \frac{(2\pi)^{\frac{1-m}{2}}}{\sqrt{m}} y^{-\frac{1}{2}\left(1-\frac{1}{m}\right)} e^{my^{1/m}} \tag{21}$$

It is well known [9] that in the n-sum in (19), the first neglected term is of the same order as the remainder R_N. It is readily verified by executing the Cauchy product that $f_k \sim O\left(\ln^k y\right)$ and, hence, negligible with respect to $I(y)$ for large y. Hence, for large y, we arrive at (21). This asymptotic expansion reduces for $m = 2$ to that of the modified Bessel functions $I_v(x) \sim \frac{e^x}{\sqrt{2\pi x}}$ for $x = 2\sqrt{y}$.

QoS Routing with Incomplete Information by Analog Computing Algorithms

J□nos Levendovszky[1], Alp□r Fancsali[1], Csaba V□gső[1], G□bor R□tv□ri[2]

Budapest University of Technology and Economics,
[1]Department of Telecommunications
levendov@hit.bme.hu, s6592fan@hszk.bme.hu,
vegsocs@hit.hit.bme.hu
[2]Department of Telecommunications and Telematics
retvari@ttt-atm.ttt.bme.hu
P□zm□ny P□ter s□t□ny 1/D., Budapest 1117, Hungary

Abstract. The paper proposes novel algorithms for Quality of Service (QoS) routing in IP networks. The new algorithms can handle incomplete information, when link measures (e.g. link delays, bandwidths ...etc.) are assumed to be random variables. Incomplete information can arise due to aggregated information in PNNI and OSPF routing protocols, which make link measures characterized by their corresponding p.d.f. It will be demonstrated that the task of QoS routing can be viewed as quadratic optimization. Therefore, neural based optimization algorithms implemented on an analog computer (CNN) can provide fast routing algorithms even in the case of incomplete information. As a result, real-time routing can be carried out to meet end-to-end QoS (such as end-to-end delay) requirements.

1 Introduction

One of the major endeavors in packet switched communication networking is to ensure QoS routing. This task boils down to select paths, which satisfy given end-to-end delay or bandwidth requirements [1,2,4,13]. As a result, QoS routing is perceived as an optimization problem to search over different quality paths and choose one for which the end-to-end QoS demands are met. Unfortunately, this problem, in general, cannot be reduced to the well-known shortest path routing (tractable by the Bellman-Ford or Dijkstra algorithms in polynomial time [13]). Furthermore, if link QoS parameters (e.g. link delay or available bandwidth) are regarded as random variables then routing can become an NP hard problem [3]. In this case the aim is to select a path which guarantees the fulfillment end-to-end QoS criteria with maximum probability. We term this type of routing as a Maximum Likely Path Selection (MLPS) procedure. In this paper, MLPS will be reduced to a quadratic optimization, which can then be carried out by an analog computer called Cellular Neural Network (CNN).
The assumption that link QoS parameters are random variables is made on the following premises: (i) information aggregation (i.e., in the case of distant network components delay information is aggregated into an average delay) as detailed in

OSPF and PNNI protocols [1,8], or (ii) randomly fluctuating delays or available bandwidths (where the current values of these parameters depend on the momentary traffic scenario [2]). In both cases, routing must be performed with incomplete information, which means that the routing algorithm is designed to select a path, which can fulfill end-to-end QoS criteria with maximal probability.

2 Routing with Incomplete Information

To model the routing problem let us assume that the following quantities are given:

- there is a graph $G(V,E)$ representing the network topology;
- each link $(u,v)\in E$ has some QoS descriptors $\delta_{(u,v)}$ (e.g. $w_{(u,v)}$ bandwidth or $\tau_{(u,v)}$ delay) which are assumed to be independent random variables subject to a probability distribution function $F_{(u,v)}(x) = P(\delta_{(u,v)} < x)$;
- there is an end-to-end QoS criterion (e.g. $\min_{(u,v)\in \mathsf{R}} \delta_{(u,v)} \geq A$ for some A in the case of bandwidth requirement or $\sum_{(u,v)\in \mathsf{R}} \delta_{(u,v)} < T$ in the case of end-to-end delay requirement), where R stands for a path connecting a predefined source node s and a destination node f;
- the objective is to find an optimal path $\tilde{\mathsf{R}}$ which most likely fulfills the given QoS criterion, namely:

$$\tilde{\mathsf{R}} : \max_{\mathsf{R}} P\left(\min_{(u,v)\in \mathsf{R}} \delta_{(u,v)} \geq A\right) \quad \text{(1a)} \quad \text{or} \quad \tilde{\mathsf{R}} : \max_{\mathsf{R}} P\left(\sum_{(u,v)\in \mathsf{R}} \delta_{(u,v)} < T\right) \quad \text{(1b)}$$

One must note that in the first case δ is said to be a "bottleneck" type of link measure, whereas in the second case δ is said to be an additive type of link measure.

The path $\tilde{\mathsf{R}}$, introduced above, will be referred to as the Most Likely Path (MLP). It is well known that Shortest Path Routing (SPR) can be solved in polynomial complexity by the Dijkstra or Bellman-Ford algorithms. Therefore, mapping an MLP problem into an SPR is equivalent with proving that MLP can be solved in polynomial time. The following lemma establishes that a bottleneck measure MLP can easily be solved by using traditional SPR algorithms.

Lemma 1: The solution of $\tilde{\mathsf{R}} : \max_{\mathsf{R}} P(\min_{(u,v)\in \mathsf{R}} \delta_{(u,v)} \geq A)$ is equivalent to solving a traditional shortest path problem with the metric assigned to the $(u,v)th$ link being $\mu(u,v) = -\log P(\delta_{(u,v)} \geq A) = -\log\{1 - F_{(u,v)}(A)\}$.

Proof: We seek the path $\tilde{\mathsf{R}} : \max_{\mathsf{R}} P(\min_{(u,v)\in \mathsf{R}} \delta_{(u,v)} \geq A)$, which is equivalent to $\tilde{\mathsf{R}}$: $\max_{\mathsf{R}} P\left(\bigcap_{(u,v)\in \mathsf{R}} \delta_{(u,v)} \geq A\right)$. Assuming independent random variables $P\left(\bigcap_{(u,v)\in \mathsf{R}} \delta_{(u,v)} \geq A\right)$

$= \prod_{(u,v)\in\mathsf{R}} P\left(\delta_{(u,v)} \geq A\right)$, one can write $\tilde{\mathsf{R}} : \min_{\mathsf{R}} \sum_{(u,v)\in\mathsf{R}} -\log P\left(\delta_{(u,v)} \geq A\right)$. Therefore assigning measure μ as $\mu_{(u,v)} := -\log P\left(\delta_{(u,v)} \geq A\right)$ MLP routing can indeed be solved by SPR.

□

However, if the link descriptor is delay, then QoS routing yields an intractable problem, as stated by the following lemma.

Lemma 2: (Gu□rin et al) The solution of $\tilde{\mathsf{R}} : \max_{\mathsf{R}} P\left(\sum_{(u,v)\in\mathsf{R}} \delta_{(u,v)} < T\right)$ (which will be referred to as Delay Problem (DP)) in general is NP hard.

The proof is based on the fact that the problem of $\tilde{\mathsf{R}} : P\left(\sum_{(u,v)\in\mathsf{R}} \delta_{(u,v)} < T\right) > \pi$ (where $0 < \pi < 1$ is some given threshold) is also intractable. For further details regarding the proof, see [3].

Therefore, our effort is focused on introducing special constraints under which the optimization problem (1b) lends itself to analytical tractability, still preserving the main attributes of the problem (i.e., yielding results which are still relevant to practical networking scenarios).

3 MLPS as a Quadratic Optimization Problem

In this section, we reduce MLPS to a quadratic optimization problem. In order to obtain a formal model let us introduce the following quantities:

- The link delay (or other QoS parameters) associated with link (u,v) is assumed to be a discrete random variable taking its values from a finite set $\tau_{(u,v)} \in \{T_1,...,T_M\} = \mathsf{T}$ with a discrete probability mass function $p_{(u,v),1},....,p_{(u,v),M}$. From this notation is clear that each link takes its delay value from the same set T, however, the probability mass function characterizing the random delay of link (u,v) can vary from link to link.
- These probability mass functions are summarized in the three-dimensional array $\overline{\overline{\overline{R}}} \to R_{ijn} = -\log P\left(\tau_{(i,j)} = T_n\right) \geq 0$. More precisely, R_{ijn} indicates the negative logarithm of the probability of taking a hop from node i to node j with delay T_n.
- The delay structure of the graph is summarized in the three-dimensional array: $\overline{\overline{\overline{D}}} \to D_{ijn} = T_n$, where the element D_{ijn} indicates that hopping from node i to node j introduces delay T_n in the path.
- A path together with its delay is represented by a three-dimensional array

$$\overline{\overline{\overline{V}}} \rightarrow V_{ijn} = \begin{cases} 1 & \text{if arriving at node } j \text{ at stage } i \text{ with delay } T_n \\ 0 & \text{otherwise} \end{cases}$$

- A Start Point Indicator Array (SPIA) and an End Point Indicator Array (EPIA) defined as

$$S_{ijn} = \begin{cases} 1 & \text{if } i = 1, j = s \\ 0 & \text{otherwise} \end{cases} \qquad E_{ijn}^{(k)} = \begin{cases} 1 & \text{if } i = k+1, j = f \\ 0 & \text{otherwise} \end{cases}$$

SPIA expresses that the path must start at node s, whereas EPIA indicates that the path ends at node f after k steps.

One must note that the three-dimensional arrays defined above have N^2M number of elements. In the forthcoming discussion we only consider the task of solving the problem of k-hop routing (finding a path containing only k links). This assumption is widely used to make the problem tractable.

3.1 Properties of a Valid Path

From the definitions given above, one can summarize the properties of a valid path as follows:

1. The two-dimensional projection V_{ijn} of a three-dimensional array $\overline{\overline{\overline{V}}}$ for a fixed n can have at most one element different from zero in each row (for any given n). This element equals 1.
2. The two-dimensional projection V_{ijn} of a three-dimensional array $\overline{\overline{\overline{V}}}$ for a fixed n can have at most one element different from zero in each column (for any given n). This element equals to 1.
3. The three-dimensional array $\overline{\overline{\overline{V}}}$ must have only $k+1$ elements equal to 1 and the remaining elements should be zero.
4. If the path should start from node s then at least one element is 1 in the "depth-row" of $\overline{\overline{\overline{V}}}$, namely V_{1sn} for some n, $n = 1,..., M$.
5. If the path should end at node f after k number hops (assuming k-hop routing), then at least one element is 1 in the "depth-row" of $\overline{\overline{\overline{V}}}$, namely $V_{k+1fn} = 1$ for some n, $n = 1,...., M$.

3.2 QoS Routing as a Constrained Quadratic Optimization Problem

As could be seen in the last section, MLPS should take place over the space of three-dimensional data arrays which fulfill Criteria 1,..,5, respectively. This space is denoted by $\mathbf{V}$. Therefore, the objective is to select a path (or a corresponding $\overline{\overline{\overline{V}}}$), for which

$$\overline{\overline{\overline{V}}}_{opt} : \min_{\overline{\overline{\overline{V}}} \in \mathbf{V}} \sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} R_{jln} V_{i+1lm} + \left(\sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} D_{jln} V_{i+1lm} - T \right) \quad (2)$$

where T indicates the end-to-end delay requirement. One can easily see that the first term in the summation is related to the probability of a path as follows:

$$\begin{aligned} \sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} R_{jln} V_{i+1lm} &= \sum_{(u,v)\in \mathsf{R}} -\log P(\tau_{(u,v)} = T_n)\Big|_{n:V_{uvn}=1} = \\ = -\log \prod_{(u,v)\in \mathsf{R}} P(\tau_{(u,v)} = T_n)\Big|_{n:V_{uvn}=1} &= -\log P\left(\bigcap_{(u,v)\in \mathsf{R}} (\tau_{(u,v)} = T_n)\Big|_{n:V_{uvn}=1} \right) \end{aligned} \quad (3)$$

To minimize (3) means that the array $\overline{\overline{\overline{V}}}$ is going to represent the most probable delay realization along the path R.

The second term $\sum_i \sum_j \sum_l \sum_n \sum_m V_{ijn} D_{jln} V_{i+1lm} - T$ expresses the QoS constraint enforced over the whole path, namely the following condition must hold: $\sum_i \sum_j \sum_l \sum_n \sum_m V_{ijn} D_{jln} V_{i+1lm} = \sum_{(u,v)\in \mathsf{R}} T_n \big|_{n:V_{uvn}=1} \le A$ if the realization of the random sequence of the link delays in the path are $\tau_{(u,v)} = T_n \big|_{n:V_{uvn}=1} \quad \forall (u,v) \in \mathsf{R}$. Therefore, minimizing the first and the second term at the same time will yield a path, which fulfills the constraint in the best manner.

Hopfield type of neural algorithms can solve quadratic optimization problems in a polynomial time over the space of binary data structures. Thus, to utilize Hopfield or CNN-based methods, one has to build in additional constraints in the goal function, which enforce the solution obtained over the full binary space to be a valid path. This can be done by defining the optimization function as follows [5]:

$$\begin{aligned} &\sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} R_{jln} V_{i+1lm} + \left(\sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} D_{jln} V_{i+1lm} - T \right) + \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{n=1}^{M} V_{ijn} S_{ijn} - 1 \right)^2 + \\ &+ \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{n=1}^{M} V_{ijn} E_{ijn}^{(k)} - 1 \right)^2 + \sum_{n=1}^{M} \sum_{l=1}^{N} \sum_{\substack{j=1 \\ j\ne i}}^{N} V_{iln} V_{jln} + \sum_{n=1}^{M} \sum_{l=1}^{N} \sum_{\substack{j=1 \\ j\ne i}}^{N} V_{lin} V_{ljn} + \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{n=1}^{M} V_{ijn} - k - 1 \right)^2 . \end{aligned} \quad (4)$$

The third and the fourth term in the summation enforce that the path starts from node s and ends at node f. The fifth and sixth term impose the constraint of orthogonal rows and columns in the array $\mathbf{V}$. Finally, the seventh term is responsible to ensure a k-hop routing allowing only $k+1$ ones in the data array $\mathbf{V}$.

The three-dimensional data array $\overline{\overline{\overline{V}}}$ can be converted into a binary vector by applying the following transformation: $y_{((i-1)N+j-1)M+n} = 2(V_{ijn} - 0.5)$. With this transformation QoS routing is reduced to a quadratic optimization over $\{-1,1\}^{N^2 M}$.

3.3 Neural Based Routing Algorithms

Since the problem has been transformed into a binary quadratic optimization, it can be solved by a corresponding Hopfield type of neural network. The state transition rule is given by the following equation:

$$y_l(k+1) = \operatorname{sgn}\left(\sum_{j=1}^{N^2 M} W_{lj} y_j(k) - b_l\right), \tag{5}$$

where the weights W_{lj} and components b_l can be calculated by a simple algorithm given as follows. We know that every given quadratic expression $\phi(\mathbf{y})$ can be written in the following general form: $\phi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T \mathbf{W}\mathbf{y} - \mathbf{b}^T \mathbf{y} + \phi_0$.

One can substitute unit vectors in $\phi(\mathbf{y})$ as follows

$$\begin{aligned} &\phi(\mathbf{e}_i) = \frac{1}{2} w_{ii} - b_i + \phi_0, \quad \phi(-\mathbf{e}_i) = \frac{1}{2} w_{ii} + b_i + \phi_0 \quad (\forall i) \\ &\phi(\mathbf{e}_i + \mathbf{e}_j) = \frac{1}{2}\left(w_{ii} + w_{jj}\right) + w_{ij} - b_i - b_j + \phi_0 \quad (\text{if } w_{ij} = w_{ji}\ \forall i, j \neq i). \end{aligned} \tag{6}$$

Combining these equations we obtains

$$\begin{aligned} b_i &= \phi(-\mathbf{e}_i) - \phi(\mathbf{e}_i) & \\ w_{ii} &= \phi(\mathbf{e}_i) + \phi(-\mathbf{e}_i) - 2\phi_0 & (i = 1, 2, \ldots, N^2 M) \\ w_{ij} &= \phi(\mathbf{e}_i + \mathbf{e}_j) - \frac{1}{2}(w_{ii} + w_{jj}) + b_i + b_j - \phi_0 & (j = 1, 2, \ldots, N^2 M; j \neq i) \end{aligned}$$

In this way, matrix **W** and vector **b** can easily be identified.
It is also noteworthy that the optimization performance of the Hopfield net can greatly be improved by adding a noise to (5), yielding

$$y_l(k+1) = \operatorname{sgn}\left(\sum_{l=1}^{N^2 M} W_{lj} y_j(k) - b_l + \nu(k)\right), \tag{7}$$

where $\nu(k)$ is a random variable subject to logistic distribution $p_\nu(x) = \left(1 + e^{-\alpha(k)x}\right)^{-1}$. It can be proven that the stationary distribution of the Markov chain generated by (7) yields the global maximum with maximal probability [7]. More precisely,

$$\mathbf{y}_{opt} = \arg\max_{\mathbf{y} \in \{-1,1\}^{N \cdot N \cdot M}} \pi(\mathbf{y}) = \arg\max_{\mathbf{y} \in \{-1,1\}^{N \cdot N \cdot M}} \mathbf{y}^T \mathbf{W}\mathbf{y} - 2\mathbf{b}^T \mathbf{y},$$

where $\pi(\mathbf{y})$ denotes the stationary distribution.

3.4 Solution by CNN

Since the problem has been transformed into a binary quadratic optimization it can be solved by a corresponding CNN. As has been pointed out in [6], the optimization can be carried out by the following differential equation:

$$\begin{aligned}\dot{x}_{ijn}(t) &= -x_{ijn}(t) + (\alpha-1)V_{ijn}(t) - \sum_{m=1}^{M}\sum_{l=1}^{N}(R_{jln}+D_{jln})V_{i+1,l,m} \\ &\quad - \sum_{m=1}^{M}\sum_{l=1}^{N}(R_{ljn}+D_{ljn})V_{i-1,l,m} - A\chi_{ijn}\left(\overset{\equiv}{V}\right) + b_{ijn}\end{aligned} \tag{8}$$

where

$$\chi_{ijn}\left(\overset{\equiv}{V}\right) = \sum_{m=1}^{M}\sum_{\substack{l=1\\ l\neq i}}^{k+1} V_{ljm} + \sum_{m=1}^{M}\sum_{\substack{q=1\\ q\neq j}}^{N} V_{iqm} + \sum_{l=1}^{k+1}\sum_{\substack{m=1\\ m\neq n}}^{M} V_{ljm} + \sum_{l=1}^{k+1}\sum_{\substack{q=1\\ q\neq j}}^{N} V_{lqn} + \sum_{q=1}^{N}\sum_{\substack{m=1\\ m\neq n}}^{M} V_{iqm} + \sum_{q=1}^{N}\sum_{\substack{l=1\\ l\neq i}}^{M} V_{lkn}.$$

The values of the constants A, α and b_{ijn} are also given in [6].

If we implemented the method on Hopfield type neural network, its recursion would stabilize in $O\left((n_{\text{nodes}}n_{\text{conn}})^2\right)$ step, where n_{nodes} denotes the number of neurons (now being N^2M) and n_{conn} represents the number of neurons which are connected maximally to one neuron (in our case it is NM multiplied by a constant). From this $O\left(N^7M^4\right)$ complexity is obtained considering that we have to run the method for k=1,2,□ ,N–1. The result seems to be disappointing. However, taking into account that we can implement the method on an analog structure which is able to run in μs range, the high complexity is not an issue.

It is also noteworthy that the optimization performance of the CNN can be greatly improved by adding a noise (Wiener process) to (8) (for further details see [6]).

Unfortunately, the present CNN-technology does not allow to directly apply (8) in the case of larger than 6-7-node networks because of the relatively complex connection structure in which remote cells are also in connection. Even now investigations are conducted in order to apply the method on a 2D-CNN which has real template structure (only the neighboring cells are connected [9,10]). Currently 64x64-cell CNN is available which can allow us to implement routing in the case of usual network size. One must note that the proposed method is suitable for the case of small networks, because of the hierarchical routing [6].

4 Performance Analysis

In this section the performances of the newly developed QoS routing algorithms are analyzed by extensive simulation. In order to compare the algorithms we introduced a performance measure for the paths starting from node s and ending at node f

denoted by $\eta(G,s,f,T)$. This is defined as the ratio of the probability that the path found by a given algorithm satisfies the end-to-end delay requirement T to that of the path found by an exhaustive search, given as follows:

$$\eta(G,s,f,T) := \frac{P\left(\sum_{(u,v)\in \mathsf{R}_{\text{found by a given algorithm}}} \tau_{(u,v)} < T\right)}{P\left(\sum_{(u,v)\in \mathsf{R}_{\text{found by exhaustive search}}} \tau_{(u,v)} < T\right)} . \tag{9}$$

However, one can calculate the average $\eta(G,s,f,T)$ for a whole graph (i.e., for each possible starting node s and for each possible ending node $f \in V$, yielding the following performance function $\eta(G(V,E),T)$:

$$\eta(G,T) := \frac{1}{|V|^2 - |V|} \sum_{s\in V} \sum_{f\in V, f\neq s} \eta(s,f,T) \ . \tag{10}$$

Furthermore, one can calculate the average performance with respect to the end-to-end QoS criterion (T) given as follows

$$\eta(G) = \frac{1}{T_{\max}} \int_0^{T_{\max}} \eta(G,T)dT \tag{11}$$

and the average performance over a set of randomly generated graphs (G)

$$\eta(T) = \frac{1}{|\mathsf{G}|} \sum_{G\in \mathsf{G}} \eta(G,T) . \tag{12}$$

In the case of a given graph this measure only depends on the value of the actual end- to-end QoS requirement. The closer this function approximates the value 1, the better the performance of the corresponding routing algorithm is.

The test graph on which the algorithms have been tested was fully connected containing 7 nodes and its topology is similar to a typical Local Exchange Network. Equidistant link scaling was used along the delay axis. The link delays were chosen as Bernoulli random variables the expected value of which fell into the middle of the interval in which the link delay is assumed to be contained.

Figure 1 shows the performance of deterministic and noisy HNN routing algorithm on the test graph. On the horizontal axis the different source node - destination node pairs are indicated, whereas the values on the vertical axis correspond to the efficiency defined by (9). The corresponding bar chart indicates the related average performance defined by (10).

Figure 2 exhibits how the efficiency depends on the QoS parameter (the overall delay *T*). On the horizontal axis the value of the QoS parameter indicated, whereas the values on the vertical axis correspond to the efficiency defined by (10). The corresponding bar chart indicates the related average performance defined by (11).

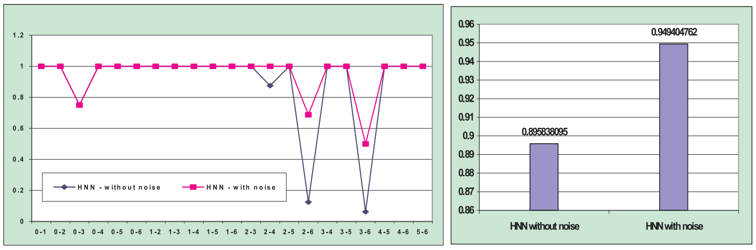


Fig. 1. The performance of HNN methods changing the end points in the test graph

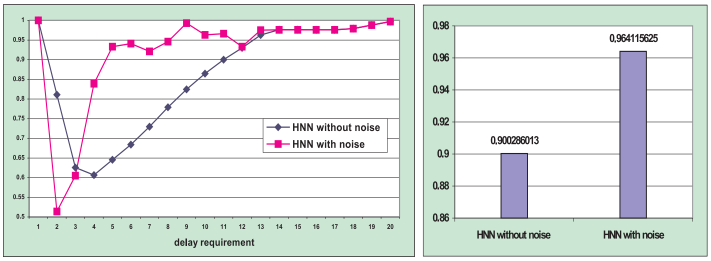


Fig. 2. The performance of HNN methods changing the QoS requirement

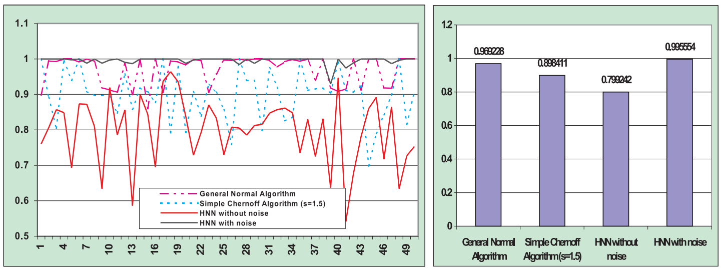


Fig. 3. The performance of HNN methods generating graphs randomly

Figure 3 shows the performance of the deterministic and noisy Hopfield QoS routing algorithm. The figure also shows the performance of two other types of methods: “General Normal Algorithm” (using Gaussian approximation to reduce the original problem to SPR) and “Simple Chernoff Algorithm” (using large deviation approach to reduce the original problem to SPR). For further details regarding the algorithms, see [4]. The SPR is solved by Bellman-Ford algorithm in these two cases. The performance measure is taken over the ensemble of 50 randomly generated 7-node graphs. Each of them was generated with probability 0.8 to have connection between each node pair. While generating networks the non-separability was ensured. Table 1 shows the distributions of link values and QoS requirements [11], which were used to obtain the results shown in Figure 3.

Table 1. Simulation parameters

Parameter	**Value**
T	Uniform distribution in $[30ms, 160ms]$.
$\delta_{(u,v)}$	Uniform distribution in $[0ms, 50ms]$.

On the horizontal axis the networks are indicated, whereas the values on the vertical axis correspond to the efficiency defined by (10). The bar chart indicates the related average performance defined by (12). It is again verified by the figure, that the routing performance of the noisy HNN is rather high, it never goes below 0.9, even over a large ensemble of network topologies.

5 Conclusion

A novel CNN-based routing algorithm was introduced in the paper, which is able to meet end-to-end QoS requirements even in the case of incomplete information. Since routing was transformed into a binary quadratic optimization, CNN based solution became available, which yielded fast and a high performance routing. The quadratic cost function (4) easily can be modified to enable searching for paths with multiple cost functions. Unfortunately, the main advantages of recently developed algorithms [11,12,14]: working in distributed manner, searching for multiple paths are not be included in this CNN based solution. Moreover, the obtainable gain considering its running time (originated from analog operations) is restricted by the network size. Assuming large network topologies other types of approximations using digital computing techniques can have shorter running time.

Acknowledgment

The research reported here was supported by Analogical and Neurocomputing Laboratory, MTA SZTAKI. The authors are grateful for the comments and guidelines they received from Prof. Tam□s Roska during their research.

References

1. Cherukuri, R., Dykeman, D.: PNNI draft specification, ATM Forum 94-0471, November 1995.
2. Lorenz, D., Orda, A.: QoS routing in networks with uncertain parameters, IEEE/ACM Trans. Networking, vol. 6., December 1998, pp. 768-778.
3. Gu□rin, R., Orda, A.: QoS routing in networks with inaccurate information: theory and algorithms, IEEE/ACM Trans. Networking, vol. 7., June 1999, pp. 350-364.
4. Levendovszky, J., R□tv□ri, G., D□vid, T., Fancsali, A., V□gső, Cs.: QoS routing in packet switched networks - novel algorithms for routing with incomplete information, Proceedings

of 9th IFIP Working Conference on Performance Modelling and Evaluation of ATM & IP Networks, Budapest, Hungary, pp. 249-260, 27-29 June, 2001.
5. Levendovszky, J., Fancsali, A., V□gső, Cs.: CNN based algorithm for QoS routing with incomplete information, 22nd Symposium on Information Theory in Benelux, Amsterdam, Netherlands, 15-16 May 2001.
6. Levendovszky, J., Fancsali, A.: Application of CNNs in modern communication technologies, Technical Report, Analogical and Neurocomputing Laboratory, MTA SZTAKI, DNS-7-2000.
7. Jeney, G., Levendovszky, J.: Stochastic Hopfield Network for Multi-user Detection, European Conf. Wireless Techn., pp. 147-150, 2000.
8. Lee, W.: Spanning tree methods for link state aggregation in large communication networks, Proc. INFOCOM, Boston, MA, April 1995.
9. Chua, L. O., Roska, T.: The CNN Paradigm, IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, (CAS-I)*, Vol.40, No. 3, pp. 147-156, 1993.
10. Chua, L. O. Yang, L.: Cellular Neural Networks: Theory, IEEE Trans. on Circuits and Systems, (CAS), Vol.35, pp. 1257-1272, 1988.
11. Chen, S., Nahrstedt, K.: Distributed QoS Routing with Imprecise State Information, Proceedings of 7th IEEE International Conference on Computer, Communications and Networks, Lafayette, LA, pp. 614-621, October 1998.
12. Chen, S., Nahrstedt, K.: On Finding Multi-Constrained Paths, Proceedings of 7th IEEE International Conference on Communications, Atalanta, GA, pp. 874-879, June 1998.
13. Chen, S., Nahrstedt, K.: An overview of quality of service routing for next generation high-speed networks: Problems and solutions, IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video*,* 12(6): 64-79, November-December 1998.
14. Sun, Q., Langendorfer, H.: A new distributed routing algorithm with end-to-end delay guarantee, IWQoS□97, May 1997.

Profile-Based Routing: A New Framework for MPLS Traffic Engineering[⋆]

Subhash Suri[1], Marcel Waldvogel[2], and Priyank Ramesh Warkhede[3]

[1] UC Santa Barbara, Santa Barbara CA 93106, USA
[2] IBM Zurich Research Lab, 8803 Rüschlikon, Switzerland
[3] Cisco Systems, San Jose CA 95134, USA

Abstract. We present a new algorithm and framework for dynamic routing of bandwidth guaranteed flows. The problem is motivated by the need to dynamically set up bandwidth guaranteed paths in carrier and ISP networks. Traditional routing algorithms such as minimum hop routing or widest path routing do not take advantage of any knowledge about the traffic distribution or ingress-egress pairs, and therefore can often lead to severe network underutilization. Our work is inspired by the recently proposed "minimum interference routing" algorithm (MIRA) of Kodialam and Lakshman, but it improves on their approach in several ways. Our main idea is to use a "traffic profile" of the network, obtained by measurements or service level agreements (SLAs), as a rough predictor of the future traffic distribution. We use this profile to solve a *multicommodity network flow* problem, whose output is used both to guide our *online* path selection algorithm as well as impose admission control. The offline multicommodity solution seems very effective at distributing the routes and avoiding bottlenecks around hot spots. In particular, our algorithm can anticipate a flow's blocking effect on groups of ingress-egress pairs, while MIRA only considers one ingress-egress pair at a time. Our simulation results show that the new algorithm outperforms shortest path, widest path, and minimum interference routing algorithms on several metrics, including the fraction of requests routed and the fraction of requested bandwidth routed. Finally, the framework is quite general and can be extended in numerous ways to accommodate a variety of traffic management priorities in the network.

1 Introduction

We present a new algorithm and framework for dynamic routing of bandwidth guaranteed flows. Our algorithm is *online*, meaning that it routes requests one at a time, without specific knowledge of future demands. We use quasi-static information about the network and traffic to select paths so as to minimize the number of requests that are rejected or the network bandwidth that is wasted. Clearly, if no assumptions are made about the flow requests, a pathologically chosen set of requests can foil *any* online algorithm. We make minimal assumptions

[⋆] Research performed at Washington University in St. Louis, partially supported by NSF grant ANI 9813723.

that are justifiable in practice and lead to significant improvement in network utilization. In particular, we assume that the ingress and egress nodes in the network are known, and that a *traffic profile* between pairs of ingress-egress nodes is also known. The traffic profile between ingress-egress node pairs can be either measured or inferred from service level agreements (SLAs). Our algorithm uses this quasi-static information in a *preprocessing* step (one multi-commodity flow computation), to determine certain bandwidth allocations on the links of the network. The online phase of the routing algorithm then routes tunnel requests using a "shortest path" (SPF) like algorithm *but with the additional information given by the preprocessing phase.* The multi-commodity preprocessing phase allows the online algorithm to exercise *admission control* by rejecting some requests because of their blocking effects in the network.

The motivation for our problem arises from the needs of service providers who must dynamically reserve bandwidth guaranteed routes in carrier and ISP networks. Following Kodialam and Lakshman [8], we will describe our algorithms in the context of setting up paths in Multi-Protocol Label Switched (MPLS) networks, although our algorithms are applicable in other contexts as well. MPLS networks [10] allow explicit routing of packets by putting labels on them, which can then be used to forward packets along specific Label Switched Paths (LSPs). Service providers can perform this encapsulation at the ingress routers, and then use LSPs to implement Virtual Private Networks (VPNs) [6] or satisfy other quality of service (QoS) agreements with clients. At the ingress routers, packet classification [9,11,12] can be used to map packets into "forwarding equivalence classes" by examining packet headers. This aggregation (mapping into equivalence classes) also has the potential advantage of smoothing out the bandwidth requirement across many bursty streams. In addition, the service providers can use a measurement-based mechanism to build a traffic profile for an ingress-egress node pair. Such a profile can be as simple as an average bandwidth requirement over a certain time period.

A Label Switched Path requires set up, meaning that all the intermediate routers between the ingress and egress nodes are specified. The path is set up using a signaling protocol such as RSVP [3] or LDP (Label Distribution Protocol [1]). The ability to specify explicit paths for any flow gives the service providers an important tool to engineer how their traffic is routed, and thereby improve the network utilization, by minimizing the number of requests that are rejected when the network becomes overloaded. Current intra-domain routing schemes, which forward packets based on destination address only, do not take into account what other flows are currently, or likely to be, requested. Thus, their routing behavior is highly myopic—they will reject a flow when the default shortest path route become congested, even if an alternative path is available. The algorithms like widest path routing also suffer from similar problems. We therefore need better schemes for routing flow requests that take better advantage of the network infrastructure, network topology, and traffic distribution. We show that this problem is NP-Complete even in highly simplified form, but propose a novel multi-commodity based framework, which eliminates many of the shortcomings of shortest path routing, widest path routing, and even minimum interference routing.

While we present our algorithm in the context of bandwidth guarantees, it can also perform routing based on other QoS metrics such as delay, loss etc. As pointed out by Kodialam and Lakshman [8], if additional constraints, such as delay or loss, are to be incorporated into SLAs, one can do so effectively by converting those requirements into a bandwidth requirement.

Our framework is quite general, and it can be extended and generalized in multiple ways to handle additional metrics and requirements. In particular, the multi-commodity flow formulation permits a *cost* function, which we minimize to achieve optimal routing. In order to minimize the number rejected requests, we use the simple *linear cost function.* A variety of *non-linear* cost functions can be used to handle features like *minimum guaranteed bandwidth* or *fairness* across multiple flows.

2 Routing Requirements

In this section, we briefly discuss the requirements that a flow routing algorithm must satisfy. Kodialam and Lakshman [8] give a detailed list of ten important criteria that a dynamic path selection algorithm must meet. We discuss only the most important requirements here.

Routing without splitting flows. It is assumed that the flow should be routed on a single path, without splitting. Many flow requests may involve traffic that is inherently unsplittable (circuit emulation or voice), and therefore it is important to route them on single paths. Thus, for each flow request, the algorithm must find a path with desired amount of bandwidth between the ingress and egress nodes, or determine that the flow is unroutable.

Online routing. We assume that the individual flow set-up requests arrive online, one at a time, and the algorithm must process each request without having to know the future requests. In network provisioning and design phase, it is customary to assume that exact point-to-point demands are known. But that assumption is highly impractical for the MPLS tunnel set up problem. While we make use of the quasi-static information such as traffic profile in our algorithm, those profiles are used only as a rough guide for the aggregate demands. Furthermore, our routing algorithm is completely online—it does not need to know anything about individual requests, their bandwidth requirements, or their time of arrival. Of course, if the actual demands in aggregate deviate significantly from the assumed profile, the performance improvement achieved by our algorithm may degrade, but that is to be expected for any online algorithm.

Computational requirement. We want the path selection algorithm to be quite fast and scalable. Individual flow set-up requests are typically processed at the ingress routers or switches, which operate at very high load and have limited computing power. Thus, the computational requirement per flow setup request must be kept as low as possible. In this regard, our algorithm is just as efficient and simple as the shortest path algorithm, and substantially faster than Kodialam-Lakshman algorithm. The expensive part of our algorithm is the preprocessing phase, which is run very infrequently and offline, only when the quasi-static information changes. The online algorithm runs a single breadth-first search algorithm, which is several *orders of magnitude faster* than the max flow computations needed by MIRA [8].

Policy constraints. A good path selection algorithm should be able to incorporate additional policy constraints. For example, a service level agreement may require avoiding links with certain loss rate. Similarly, SLAs may require a minimum flow acceptance guarantee; for example, over a period of one hour, flows with total bandwidth at least 100 Mbps must be accepted. In Section 9, we describe mechanisms to implement policy constraints into the framework.

Traffic profile. Our algorithm uses information about "expected" flows between some ingress-egress nodes. We explain the exact form of this information later, but briefly speaking our belief is that yesterday's traffic between an ingress-egress pair can serve as a good predictor for today's traffic. This should be especially true in light of fact that service providers aggregate a large number of flows, using forwarding equivalence classes, for the ingress-egress pairs. Service providers can have multiple classes per ingress-egress pair, and keep separate profiles for various classes. These profiles can be either measurement based, or they can be inferred from service level agreements.

Routing information. Finally, like shortest path routing, our algorithm also uses only the link-state information and, like the widest path routing algorithm, it uses some auxiliary capacity information. In order to keep the presentation simple, we describe our algorithm for the centralized route serve model, though it can also be implemented in the distributed mode.

3 Review of Existing Algorithms

The most commonly used algorithm for routing LSPs is the *shortest path routing*. In the shortest path routing, the path with the least number of links between ingress and egress nodes is chosen. The routing algorithm keeps track of the current *residual capacity* for each link, and only those links that have sufficient residual capacity for the new flow are considered. The shortest path algorithm is very simple, but it can also create bottlenecks for future flows, and lead to severe network under-utilization. (See examples in Section 5.) Our new proposed algorithm is just as efficient and fast as the shortest path algorithm (during the path selection phase), but by using additional information about the network and traffic in a preprocessing phase, we can significantly reduce the number of requests that might be rejected due to inappropriate route selection. Instead of a full-fledged shortest path algorithm that has to deal with weights, which have undergone heavy manual tuning by the network operators to achieve just the desired traffic distribution, our algorithm could even use the simpler "minimum hop" algorithm (which is just a breadth-first search) to select a path in the online phase, thanks to the powerful preprocessing).

Guerin et al [7] propose a variant of the shortest path algorithm, called widest shortest path (WSP), where they choose a feasible shortest path path that has the largest residual capacity—in other words, the smallest link residual capacity along the path is maximized. While WSP certainly improves on the shortest path routing, it also has a myopic behavior—since the algorithm does not make use of the ingress-egress pairs or the traffic characteristics, it can create bottlenecks. More significantly, neither the shortest path nor the widest shortest path routing algorithm impose any form of *admission control*. Thus, these algorithms will always accept a flow if there is a feasible path in the network, even if accepting that flow has the *potential to block off a large number of future flows.* The example in Figure 2 dramatically illustrates the effect of admission control—without admission control, one can force any online algorithm to achieve close to zero network utilization!

The work most closely related to ours, and indeed the basis for our work, is the "minimum interference routing algorithm" (MIRA) of Kodialam and Lakshman [8]. MIRA is quite a bit more sophisticated algorithm than either shortest path or WSP, and it takes critical advantage of ingress-egress pairs. The basic observation in [8] is that routing a flow along a path can reduce the *maximum permissible flow* between some other ingress-egress pairs. They call this phenomenon "interference." Their thesis is that if paths that reduce a large amount of possible max-flow between other ingress-egress pairs are avoided, creation of bottlenecks can also be avoided. Their algorithm performs multiple max-flow computations to determine the path of least interference.

The idea of minimizing interference is a good one, but we believe it has several limitations. First and foremost is the observation that MIRA focuses exclusively on the interference effect on *single* ingress-egress pairs. It is not able to estimate the bottleneck created on links that are critical for *clusters* of nodes. (See examples in Section 5.) Second, MIRA considers simply the *reduction* in the maximum flow between a pair, without regard to the expected bandwidth between that pair. Thus, MIRA might reject a flow request even though the network retained sufficient residual bandwidth to route the flow between the affected pair. Finally, MIRA is computationally very expensive. While shortest path, widest shortest path, and our new proposed algorithm all perform a single shortest path computation to route a request, MIRA performs *hundreds of maximum flow computations*, each of which is several orders of magnitude more expensive than the shortest path calculation.

4 Problem Statement

We model the network as a graph $G = (V, E)$, where V is the set of routers and E is the set of links. The current *residual* capacity of a link $e \in E$ is denoted cap(e)—this is the additional bandwidth that can be routed on link e. A subset of routers are assumed to be ingress-egress routers, between which

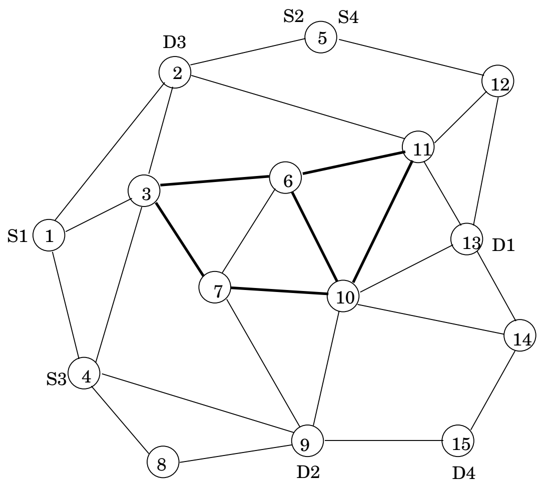


Fig. 1. An example network, showing ingress-egress nodes. This network borrowed from [8] is referred to as KL-graph in our paper.

label switched paths (LSPs) can be set up. We assume that the ingress-egress pairs are known, and that this information is quasi-static, meaning it changes very infrequently. An example is shown in Figure 1, which is borrowed from Kodialam-Lakshman [8]. We call this network the KL-graph, and it is one of the several networks on which we report our simulation results.

A request for an LSP setup is defined by a quadruple (id, s_i, d_i, b_i), where id is the request ID, s_i is the ingress (source) router, d_i is the egress (destination) router, and b_i is the bandwidth requested for the LSP. (The reason for having a separate id for each request is that there can be multiple request for the same (s_i, d_i) pair.) As mentioned earlier, all QoS requirements for the flow are assumed to have been folded into the bandwidth b_i. Given a request (id, s_i, d_i, b_i), the algorithm can either accept it, in which case it must find a path in the network from s_i to d_i along which each link has residual capacity at least b_i, or the algorithm may reject the request. The admission control feature allows our algorithm to reject a request even if there is a feasible path—this may happen if the algorithm determines that accepting this request may create a significant bottleneck for future requests (based on its knowledge of the ingress-egress pairs and their traffic profile). We assume that all LSP set up requests arrive online, one at a time, and the algorithm does not know anything about individual future requests, their bandwidth requirements, or their time of arrival.

The traffic profile information used by our algorithm records the expected flow between pairs of ingress-egress routers, and represents an aggregated demand profile between ingress-egress pairs. Such information can be either measurement-based or it can be calculated from SLAs that have been entered by a service provider with its clients. Each traffic profile is also defined by a quadruple: $(classID, s_i, d_i, B_i)$, where $classID$ is the traffic class, s_i, d_i are the ingress and egress nodes, and B_i is the aggregate traffic to be expected for this class between s_i and d_i. Between the same s_i, d_i pair, there can be multiple traffic classes (corresponding to different service types offered by the provider). Each LSP request can be mapped to a unique traffic profile class. (Conversely, a traffic profile class acts as an aggregate proxy for all the LSP requests mapped to it.)

The traffic profile is a rough indication of the amount of traffic that can be expected between a pair; the LSP set up request sequence however arrives online. A convenient way to think about this is that total sum of all LSP requests between s_i and d_i for the class i is a *random variable* with mean B_i. But the time of arrival of individual requests and their bandwidth requirements are entirely unpredictable. Thus, as for as our routing algorithm is concerned, the request sequence is completely online.

For simplicity, we assume there is a route server that knows the current network topology and available link capacities. Instead of dedicating a single machine to perform route computations, this job could also be shared among all ingress nodes without changes to the framework.

5 Examples Illustrating Limitations of Existing Routing Algorithms

In this section, we informally describe the shortcomings of existing routing algorithms using some simple illustrative examples. Our basic theme is that algorithms that do not adapt to the traffic distribution in the network (taking advantage of ingress-egress pairs and some rough estimate of the traffic flow between pairs) will always lead to suboptimal network utilization, which can be quite severe in some cases. In particular, the routing by algorithms like shortest path and WSP that do not impose any form of admission control can occasionally lead to significant bottlenecks. Simply having more information about the network or traffic does not guarantee better routing. Our proposed framework assumes minimal information about the network and traffic, which we believe can be easily obtained. Our algorithm, though as simple and computationally efficient as shortest path, leads to fewer rejected requests and better network utilization.

We use three simple examples to illustrate the shortcomings of existing routing algorithms. In order to drive home the point, these examples are necessarily artificial looking, but their general form is not at all unusual. In fact, real networks are quite likely to contains subgraphs resembling the *concentrator* or the *distributor* example. The parking lot topology is common as well, but also depends on the selection of ingress-egress pairs. Since pair selection is often outside the influence of the ISP, the occurrence of this pathological case is likely to appear in the real world.

Parking Lot. Figure 2 shows a simple network with $3n+3$ nodes. The ingress-egress pairs for the LSP set up requests are $(S_0, D_0), (S_1, D_1), \ldots, (S_n, D_n)$, and the bandwidth requested for each LSP is 1. All link capacities in the network are either 1 or $1+\varepsilon$, as shown.

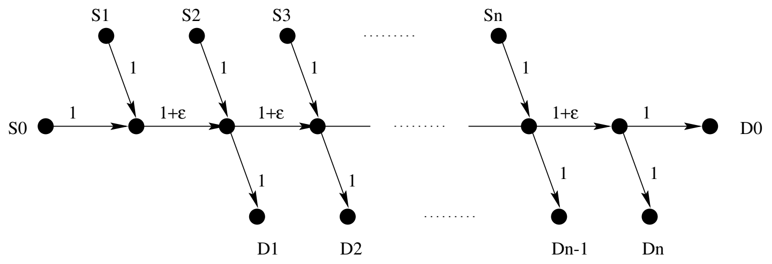


Fig. 2. The **parking lot** topology PL.

Suppose the online sequence of LSP requests arrive in the order (S_0, D_0), (S_1, D_1), ..., (S_n, D_n). Accepting the request (S_0, D_0) completely chokes off the networks—no other LSP request can be satisfied. However, since neither the shortest path nor WSP rejects flow requests if there is a feasible path, they will accept (S_0, D_0), resulting in the total network utilization of 1. An optimal algorithm will reject (S_0, D_0), and will accept $(S_1, D_1), \ldots, (S_n, D_n)$, for a total network utilization of n.

The choice of capacity $1 + \varepsilon$ for the links along the spine of the parking lot also foils MIRA—since these links are not in the min cut for any (S_i, D_i) pair, and are not considered critical. Thus, MIRA also accepts the first flow request, and ends up rejecting all other requests.

Although the links are drawn as directed, path selection and blocking behavior would remain the same for bidirectional links. In the following two examples, some of the links need to be unidirectional. Even though unidirectional links are rare (i.e., satellite downlinks and downstream-only cable modem installations), unidirectional remaining capacity is quite common. Due to asymmetric links or loads, the remaining capacity in the opposite direction could become too small to be useful.

Concentrator. Figure 3 shows a network, which we call a *concentrator graph*—one node C acts as a feeder for n ingress nodes S_1, ..., S_n. The concentrator node C is connected to a high capacity link, *fat pipe*, of capacity $n+1$, whose other endpoint is a egress node D. One high bandwidth ingress S_0 is also connected to the concentrator, through a capacity n link. S_0 is also connected to D via an alternative 3-hop path, of capacity n.

In this example, an online sequence of $n+1$ requests arrive (S_0, D), (S_1, D), ..., (S_n, D). The first request has bandwidth requirement n, while all others have bandwidth requirement 1. Using either the shortest path or the WSP, one would route the first request through the concentrator node (using 2 hops). This leaves residual capacity 1 along the link CD, and so of the remaining n requests at most one can be satisfied.

This example also illustrates the shortcoming of MIRA—the fat link CD is not in the minimum cut for any *individual* ingress-egress pair. Thus saturating it doesn't seem harmful to MIRA. So, the MIRA algorithm will also choose incorrect paths in this scenario. The optimal algorithm will route the

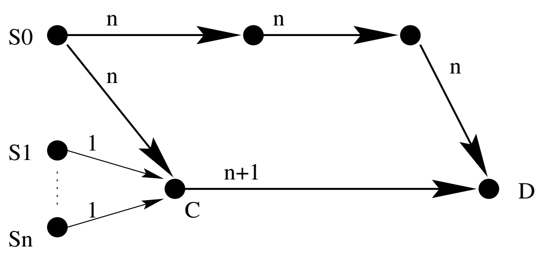


Fig. 3. The concentrator topology CN.

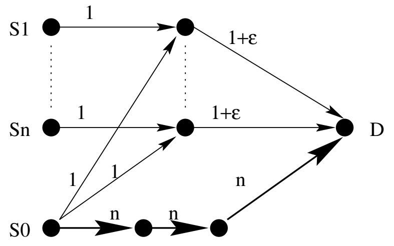


Fig. 4. The distributor topology DS.

(S_0, D) request along the top alternative path, and use the fat link to route the n 1-unit requests from S_i to D.

Distributor. While the preceding example shows why it may be a good idea to not use the fat pipe sometimes, our next example shows that the converse is also true.

In this example, we get n requests between S_0 and D, each of bandwidth 1. In addition, we also get n requests between each S_i and D, also of bandwidth 1. Again, the shortest path algorithms (shortest path and WSP) will use the two-hop paths, for each of the first n requests, choking off the $1 + \varepsilon$ links. Thus, each of the remaining n requests between S_i and D are rejected. The routes selected by MIRA are also the same, since the links of capacity $1 + \varepsilon$ are not in the minimum cut for any S_i, D pair. By contrast, the optimal algorithm will route all the requests from S_0 along the bottom fat path (3 hops), leaving the top 2-hop paths for S_i to D requests.

The preceding examples are meant to illustrate how a bad path selection for one flow can create significant bottlenecks for future flows. An online routing algorithm that does not have any additional information about the flows can perform quite poorly in the worst case. As the parking lot example shows, in some cases these algorithms cannot guarantee that even 1% of the network bandwidth is utilized, whereas the optimal algorithm achieves 100%. We build on the work by Kodialam and Lakshman [8] and propose a new algorithm as well as general framework, where we exploit information about the ingress-egress nodes as well as a measured (or estimated) traffic profile to perform both path selection and admission control. Our algorithm is both simpler than MIRA and it also performs better in many cases where MIRA falls into the same traps as the shortest path or widest shortest path routing algorithms.

6 Multi-commodity Flows

We begin with the observation that even if the exact sequence of tunnel requests were known in advance, the problem is intractable. In particular, given an offline sequence of LSP set up requests, it is NP-Complete to determine what is the maximum number of requests that can be simultaneously routed. Thus, the difficulty is not necessarily in the online nature of the problem—rather it lies in having to choose which of many paths to select for routing a flow. We turn this difficulty around by formulating the offline problem as a *multi-commodity flow problem*, on a modified network. We use the traffic profile data for the ingress-egress pairs as the offline aggregate data. The solution to the multicommodity flow problem is then used to *pre-allocate* link capacities to various flows, which are then used by the online algorithm to perform path selection. When the allocated capacity for a flow becomes zero (or was assigned zero from the beginning), that flow request is rejected—even though there might be sufficient capacity in the network to route that flow. Let us begin with some preliminaries about multicommodity flows. Interested reader can find a comprehensive treatment of network flows in the book [2].

Given a directed graph $G = (V, E)$, with positive capacity $\text{cap}(u, v)$ for each edge (u, v), a *flow* on G is a real-valued function f on node-pairs having the following properties:

Skew Symmetry. $f(v, w) = -f(w, v)$. If $f(v, w) > 0$, then we there is a flow from v to w.

Capacity Constraint. $f(v, w) \leq \text{cap}(v, w)$. If (v, w) is not an edge of G, then we assume that $\text{cap}(v, w) = 0$.

Flow Conservation. For every vertex v, other than the source or the sink (i.e., ingress or egress), the flow is conserved: $\sum_w f(v, w) = 0$.

It is straightforward to prove that the problem of determining whether a given (offline) set of LSP requests can be routed is NP-Complete.

Theorem 1. *Given a network $G = (V, E)$, where each link has a positive capacity, and a set of k LSP requests (id, s_i, d_i, b_i), for $i = 1, 2, \ldots, k$, deciding whether it is possible to simultaneously route all k requests in G is* NP-Complete.

Indeed, the LSP routing problem is a generalization of the simple two-commodity integral flow problem, which is known to be NP-Complete [5]. The 2-commodity integral flow problem asks whether it is possible to find two flow functions that deliver some required set of flows from two source nodes to two sink nodes. Specifically, suppose we are given a directed graph $G = (V, E)$, node pairs (s_1, d_1), (s_2, d_2), positive integral capacity $\text{cap}(e)$ for each edge $e \in E$, and bandwidth requirements b_1 and b_2. Then, it is NP-Complete to decide if there are flow functions f_1, f_2 such that (1) for each link $e \in E$, $f_1(e) + f_2(e) \leq \text{cap}(e)$,

(2) for each node other than s_1, s_2, d_1, d_2, flows f_1 and f_2 are conserved, and (3) the net flow to d_i under f_i is at least b_i.

We are now ready to describe the details of our algorithm.

7 Profile-Based Routing

Examining the problem more closely, we find that the intractability of LSP set up problem stems from two requirements: unsplittability of the flows, and separate demand functions for each flow. In other words, if flows are allowed to be split, and if the objective is to maximize *total* flow rather than to satisfy each individual flow, then the problem can be solved efficiently through linear programming. Unfortunately, in the LSP problem, we do not want flows to be split, and we do want to enforce some kind of fairness so as to admit as many flow as possible. Fortunately, we are able to finesse the problem on both counts by using a multi-commodity flow framework on the *traffic profiles*, rather than individual flows. First, the individual flow requests sizes are typically much smaller than the link capacities—for instance, the link capacities might be range from OC-12 to OC-192, while a typical request might be just a few megabits per second. Second, we use the multicommodity flow in the preprocessing phase, where "commodities" correspond to highly aggregated traffic profiles, and not individual LSP requests. So, when a commodity is split, it does not mean that a flow is split; rather it just means that a "group" of flows is routed on a different path than another group. An individual LSP request is never split—our algorithm either finds a single path to route it, or reject it.

Our algorithm has two phases: a preprocessing phase, where we solve a multicommodity flow problem to pre-allocate link capacities for various traffic classes; and an online routing phase, where each LSP request is routed online using a shortest path like algorithm. Let us first describe the preprocessing phase.

7.1 Multi-commodity Flow Preprocessing

The input to the preprocessing phase is the network $G = (V, E)$, with capacity $\text{cap}(e)$ for each edge $e \in E$. We are given a set of traffic profiles $(classID, s_i, d_i, B_i)$, where $classID$ is the traffic class, s_i, d_i are the ingress and egress nodes, and B_i is the aggregate bandwidth requirement for this class between s_i and d_i. We treat each traffic class as a separate *commodity*. Suppose there are k commodities, numbered 1 through k. The goal is to find routes in the network to send as much of each commodity as possible from its source node to the destination node.

As noted earlier, satisfying all bandwidth requirements however may not be possible. We therefore put additional edges in the network, called *excess edges*, so that the problem always have a feasible solution, and use *edge costs* to distinguish between the network edges and the excess edges. In particular, we add an **infinite capacity** excess edge between each ingress-egress pair, as shown in Figure 5. Thus, $cost(e) = 1$ if $e \in E$, and $cost(e) = \text{cap}(e) = \infty$ if e is

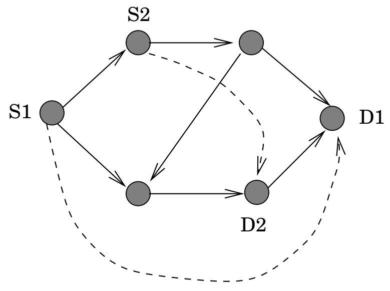


Fig. 5. The excess edges added to make the multicommodity flow always feasible. The cost of each excess edge is ∞, a large constant, while all other edges have cost one.

an excess edge, where ∞ is an appropriately large number. The large cost of the excess edges forces as much of the feasible flow as possible to go through original network edges. Let G' denote the graph obtained by adding these excess edges.

Now, let $x_i(e)$ denote a real-valued variable, denoting the amount of commodity i that is routed through edge e. Then, the multicommodity problem to be solved for graph G' is to

$$\text{minimize} \quad \sum \left(cost(e) \sum_{i=1}^{k} x_i(e) \right)$$

subject to the following constraints:

– capacity constraints are satisfied for all edges—if e is not an excess edge, then $\sum_{i=1}^{k} x_i(e) \leq \text{cap}(e)$,
– the flow for each commodity is conserved at all nodes, except the corresponding ingress and egress nodes,
– the amount of commodity i reaching its destination, d_i, is B_i.

The output of the multicommodity flow computation is the values for the variables $x_i(e)$. We use these values to set a *pre-allocation* of e's capacity for various flows. In other words, $x_i(e)$ part of e's capacity will be used by the online algorithm to route flows belonging to the traffic class i. In summary, the multi-commodity phase of the algorithm determines admission control thresholds for each traffic class, and computes pre-allocation of link capacities to maximize network utilization. The online routing phase of the algorithm is described next.

7.2 Online Path Selection for LSP Requests

The input to this phase of the algorithm is the input graph $G = (V, E)$, where for each edge $e \in E$, we keep track of the *residual capacity* $r_j(e)$ for each traffic class $j = 1, 2, \ldots, k$. (Note than these residual capacities are per traffic class, not per flow.) The initial value for $r_j(e)$ is set to $x_j(e)$, which is the output of the multi-commodity preprocessing phase. The algorithm then process an online sequence of LSP set up requests (id, s_i, d_i, b_i), where id is the request ID, s_i is the ingress (source) router, d_i is the egress (destination) router, and b_i is the bandwidth requested for the LSP. We assume that each LSP can be mapped (by the ingress router s_i) to a unique traffic class. Our online routing algorithm runs on the reduced graph, which uses the pre-allocated capacities corresponding to this class. In this reduced graph, we select a minimum hop path between s_i and d_i, if one exists.

PROFILE BASED ROUTING

Input: The input graph $G = (V, E)$. For each edge e, we maintain residual capacity $r_j(e)$ for each commodity (traffic class) $j = 1, 2, \ldots, k$. The LSP request is between an ingress-egress pair s, d, and the bandwidth requirement is b. Let j be the traffic class to which this LSP belongs.

Output: A path from s and d, such that for each edge e along this path there had been $r_j(e) \geq b$ (during the algorithm, $r_j(e)$ is updated to contain the updated residual bandwidth).

Algorithm:

1. Delete from G all edges e for which $r_j(e) < b$. (These edges have insufficient residual capacity for class j.)
2. In the reduced graph, find a path P with minimum number of hops, using a breadth first search, from s to d.
3. For each edge e in the path P, decrease the residual capacity $r_j(e)$ by b.
4. Route LSP (s, d, b) along the path P.

7.3 Complexity Analysis

If the network has N nodes and M edges, the breadth first search algorithm computes a shortest path in $O(N + M)$ time. This is a linear-time algorithm, and should be several orders of magnitude faster than the MIRA algorithm, which needs to perform several hundred (as many as the number of ingress-egress pairs) maxflow computations. *Each maxflow computation itself takes $O(N^3)$ time.* Thus, during path selection phase, our algorithm has the same run time complexity as the currently used shortest path algorithm. Our algorithm is faster than the widest shortest path routing algorithm, because that algorithm must execute a Dijkstra style shortest path computation.

The preprocessing phase of our algorithm solves a minimum cost multi-commodity flow problem, which can be slow. But that step can be executed offline, and does not require recomputation unless the network information changes, such as ingress-egress pairs or their traffic profile. Those changes are very infrequent. Thus, our algorithm needs occasional heavy preprocessing to build a pre-allocation table, which it uses to run the online path selection phase.

Table 1. Worst-case performance improvement.

Graph Name	Total Req.	Requests Routed by SPF	MIRA	PBR	Factor of Improv.
PL	$1+n$	1	1	n	n
CN	$2n$	n	n	$2n$	2
DS	$2n$	n	n	$2n$	2

8 Performance Results

Without real network topologies and large amounts of traffic data, it is difficult to perform meaningful and conclusive experiments. We will follow the tradition set by other authors, and perform experiments on several hand-crafted topologies, using both worst-case and synthetic flow data. We present qualitative as well as quantitative evidence for why we believe our Profile-Based Routing algorithm should (and does) perform better than others. One very attractive feature of our algorithm is that it is computationally as efficient as the shortest path or widest path routing, and substantially faster than MIRA.

We used four network topologies to measure the performance of our Profile-Based Routing (PBR) algorithm. The first three topologies are the ones we used for illustration in Section 5. The fourth topology, called KL, is the one used by Kodialam and Lakshman [8] in their experiments. In the parking lot topology (PL), all link capacities are set to 4800 (to model OC-48). In the concentrator (CN) and distributor (DS) topologies, we used $n = 5$, and scaled up all link capacities by 800. Thus, all links with capacity 1 in Figure 3 and capacity $1+\varepsilon$ in Figure 4 become links of capacity 800, while those with capacity n or $n+1$ become links of capacity 4800. In the network KL, all light edges have capacity 12, while dark ones have capacity 48 (meant to model OC-12 and OC-48 links, respectively). In their paper, Kodialam and Lakshman also used a scaled-up version of their network, in which capacity of links 2–3, 2–5, and 14–15 is increased to 48, and then all capacities are multiplied by 100. Finally, we used a publically available implementation of the minimum cost multi-commodity flow algorithm, PPRN package, for our preprocessing phase (available at `http://www-eio.upc.es/~jcastro/pprn.html`).

8.1 Worst-Case Results

We did not have access to an implementation of MIRA or WSP for our studies, so we compared the performance of our algorithm with the shortest path routing. In the absence of those implementations, we were also unable to compare worst-case performance of those algorithms. We can, however, infer their behavior on the three constructed network topologies, namely, the parking lot (PL), the concentrator (CN) and the distributor (DS). Table 1 documents these results.

In the parking lot topology (PL), if the first request is between the nodes S_0 and D_0, then all three algorithms (shortest path, WSP, and MIRA) accept it, which blocks all future requests from being routed. Our new algorithm rejects the first request, and is then able to satisfy all remaining n requests between S_i and D_i. As the number of ingress-egress pairs n increases, the percentage of network utilized by shortest path, WSP, or MIRA goes to zero.

Kodialam and Lakshman also propose a cost threshold (total weight $< W$) for admission control. However, that modification of MIRA also doesn't work for this topology since none of the edges used by the first path from S_0 to D_0 are in the minimum cut of any S_i, D_i pair, and consequently the weights of these edges remain zero.

In the concentrator topology (CN), a single request of size n by source S_0 will be routed by both shortest path and MIRA along the path that goes through the concentrator node C, which then blocks all future requests between S_i and D. In this case, the edge CD is not found to be critical by MIRA because it does not belong to the minimum cut of any single ingress-egress pair; it is only in the minimum cut for a cluster of ingress-egress pairs. Thus, in this case, PBR routes all $2n$ units of traffic, while the other three algorithm route only n units.

The same performance is also observed in the distributor topology (DS).

8.2 Simulation Results

We next carried out a series of experiments to measure the performance of PBR relative to the shortest path algorithm, using randomly generated request sequence.[1] In each experiment we generated a random sequence of individual flow requests, and measured the performance of our profile based routing as well as the minimum hop routing. The performance was measured both in terms of the number of flows routed, as well as the total bandwidth request that was satisfied.

Following Kodialam and Lakshman [8], we varied the bandwidths requested by individual flows between 1 and 4. This was intended to capture the fact that individual flow requests are much smaller than link capacities. However, we also ran tests to evaluate the effect of larger individual flow bandwidths.

The individual flow requests are generated in proportion to their traffic profile data. That is, if a flow belongs to traffic class i, and the total bandwidth of class i is B_i, then a flow from this class was generated with probability $B_i / \sum_j B_j$. In some cases, the traffic profile data was generated manually. In others, we used the multi-commodity flow algorithm to find feasible aggregated flows, which were then used as the profile data. Because of the random process, the expected amount of traffic requested for a traffic class was a random variable, with mean set to the profile value of that class.

[1] We would have liked to also present results from a performance comparison between PBR and MIRA. Unfortunately, we were neither able to get access to the MIRA implementation to run it on different topologies, nor were we able to obtain the simulation parameters used in [8], so that we could run our experiments under the same set of parameters.

Table 2. Total bandwidth routed by the shortest path algorithm vs. PBR.

Graph	SPF	PBR	Improvement
CN	$7,323$	$8,799$	20.16%
DS	$6,474$	$7,200$	11.21%
KL	$7,913$	$8,400$	6.15%
KL2	$7,863$	$8,400$	6.83%
PL	$14,686$	$23,999$	63.41%

Table 3. Total number of requests routed by the shortest path algorithm vs. PBR.

Graph	SPF	PBR	Improvement
CN	2,921	3,509	20.13%
DS	2,616	2,896	10.70%
KL	3,193	3,392	6.23%
KL2	3,137	3,355	6.95%
PL	6,017	9,570	59.05%

8.3 SPF Routing vs. PBR

Clearly, when the network is lightly loaded, the shortest path routing, or any other routing, is expected to do well. The beneficial effects of admission control and good path selection become evident only when the network is at least partially congested. Towards this end, we first generated enough requests that almost all paths between all ingress-egress pairs were saturated. The number of requests satisfied at this point is used as an indication of how important path selection and admission control are in improving the network utilization. In each experiment, the results are averaged over several runs so as to smooth out any effects of random request generation.

Table 2 shows the results of this experiment, by measuring the total amount of bandwidth that is routed by the two algorithms.

Table 3 shows the results of the same experiment, but this time measures the total number of flow requests that were accepted by the two algorithms. Thus, even for random requests, the PBR seems to consistently outperform the shortest path algorithm.

Our next experiment tried to evaluate the effect of increasing the size of the maximum bandwidth requested by an individual flow. In this experiment, the individual flow requests were generated with a random bandwidth requirement in the range from 1 to 48 (that is, the largest request size being 1% of the link capacity). The number of flows was proportionately reduced to keep the total bandwidth requested the same. The increased bandwidth size didn't make much difference. Tables 4 and 5 show these results.

The number of flow set-up requests satisfied by both algorithms are compared in Table 5. Fragmentation of edge capacities does not seem to have any significant

Table 4. Total bandwidth routed: shortest path vs. PBR when the largest bandwidth requested by an individual flow is 48.

Graph	SPF	PBR	Improvement
CN	7,463	8,757	17.34%
DS	6,570	7,153	8.87%
KL	7,837	8,396	7.14%
KL2	7,907	8,399	6.22%
PL	15,319	23,991	56.61%

Table 5. Total number of requests routed by shortest path algorithm vs. PBR when the largest bandwidth requested by an individual flow is 48.

Graph	SPF	PBR	Improvement
CN	313	367	17.13%
DS	276	301	9.04%
KL	322	347	7.64%
KL2	334	354	6.08%
PL	655	990	50.99%

impact. The improvement shows a similar form as for smaller requests, as seen in above tables. The issue of fragmentation of edge capacities is discussed in detail later.

8.4 Curse of Bandwidth Fragmentation

Assuming that the network traffic shows short or long-term persistent pattern, the profile data should be a good predictor of the actual observed traffic. Let us suppose for a minute that the flow requests closely matched the traffic profile used by the algorithm. How close is the performance of PBR compared to an optimal (but offline) algorithm? The multi-commodity flow solution gives an *upper bound* on how much flow is routable, which may not be achieved by our online algorithm due to *bandwidth fragmentation.* The pre-allocations $x_i(e)$ are portions of each link that are reserved for a traffic class, but because the individual flows have arbitrary bandwidth requirements, we may end up reserved capacities on different links that are insufficient to route an individual request without splitting it. How severe is the effect of this bandwidth fragmentation? A measurement of this may indicate *how close to optimal* does PBR come?

In this experiment, we generated individual requests with random bandwidth requirements in the range from 1 to max. Four values of max, namely, 4, 12, 48 and 192, were used, and for each value, the total number of requests generated was scaled down so as to keep total bandwidth requested about the same. Table 6 shows these results. As expected, the amount of bandwidth potentially wasted due to fragmentation increases with larger requests, but the degradation is quite

Table 6. Fraction of link capacities that can be potentially wasted due to fragmentation, as a function of the maximum size of individual requests. The percentage waste is measured as the reduced amount of total bandwidth routed as compared to the optimal.

	Max. Flow Size			
Graph	4	12	48	192
CN	0	0.10	0.49	1.32
DS	0	0.26	0.26	3.57
KL	0	0	0.012	0.43
KL2	0	0	0.01	0.51
PL	0	0.01	0.03	0.38

small for even very large individual flow requests (4% of the link capacity). Even then, there remains a substantial net gain for Profile-Based Routing in all the scenarios.

8.5 To Accept or Not to Accept?

PBR imposes an admission control and judiciously rejects flow requests that might create significant bottlenecks in the network. The obvious hope (and expectation) is that the flows whose bottleneck we are trying to avoid will eventually be requested. What happens if those flows are not requested? In that case, PBR might have a lower network utilization that a more myopic routing algorithm, such as the shortest path algorithm. In order to evaluate this aspect of PBR, we decided to take performance snapshots at intermittent times during the online run of the algorithm. In other words, we measured what fraction of the incoming requests were accepted, at intervals of $10\%, 30\%, 50\%, 70\%, 90\%$ of the total traffic. If PBR aggressively imposes admission control in the beginning and saves network resources for flows that do not arrive for a long time, its performance should be lower in the early duration of the run. As Table 7 below shows that it does not take long for the admission control of PBR to pay off. Except for one minuscule drop in the KL topology, the number of requests accepted by PBR always seems to be more than the number accepted by the shortest path algorithm.

9 Concluding Remarks and Extensions

We presented a new Profile-Based Routing algorithm for dynamic routing of bandwidth guaranteed paths. The online routing phase of the algorithm is as simple and computationally efficient as the commonly used min hop routing or the widest shortest path routing, and it is substantially faster than the recently proposed minimum interference routing algorithm [8]. Our algorithm improves network utilization, and accepts more flows than these other algorithms, because of its improved path selection and admission control. The algorithm takes advantage of quasi-static information about the network and traffic in an offline

Table 7. Performance improvement of PBR over the shortest path algorithm at various points during the run of the algorithm.

	Fraction of requests				
Graph	0.1	0.3	0.5	0.7	0.9
CN	0	0	0	15.03	25.79
DS	0	2.41	6.82	14.22	20.14
PL	0	0	0	23.02	62.32
KL	0	0	-0.38	8.54	6.34
KL2	0	0	0	6.11	7.19

preprocessing phase, whose output is used to both guide our *online* path selection algorithm as well as impose admission control. In particular, our algorithm is able to spot potential bottleneck links that may be in the min cut of "clusters" of ingress-egress pairs (cf. concentrator and distributor topologies), as opposed to single pairs identified by MIRA. We were unable to perform a direct comparison to MIRA, as not enough information was available.

The multi-commodity preprocessing framework proposed in our paper is quite powerful and admits numerous extensions and generalization, which can be used to implement additional policies and requirements. Just to illustrate the ideas, we mention two such extensions.

Minimum Service Level. Suppose a service provider wants to ensure that a bursty traffic class receives at least a guaranteed minimum level of service. In other words, while the expected bandwidth of a traffic class i might be B_i, the service provider wants to ensure that at least M_i level of bandwidth is guaranteed during a certain time period. We can implement this requirement by using a different *cost function* in the objective function of our multi-commodity formulation. The objective function is augmented by an additive term C which kicks in if more than $B_i - M_i$ units are routed along the *excess edge* corresponding to this traffic class.

Imposing Fairness. A flow routing algorithm can achieve large network utilization, but may unfairly punish some clients by rejecting a disproportionate share of their flows. Service providers can implement a minimum level of fairness by ensuring that a given set of traffic classes each receives a proportionate share of total bandwidth. For a traffic class j, we add $\alpha^{B_j - M_j}$ to our objective function, which is exponential in the bandwidth not routed, for a tunable parameter α. This guarantees that one class receiving an unfairly low bandwidth allocation leads to a steep cost, and will be avoided.

References

1. L. Andersson, et al. LDP Specification. Internet RFC 3036, 2001.

2. R. K. Ahuja, T. L. Magnanti and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.
3. R. Braden, D. Clark, and S. Shenker. RSVP: A New Resource ReSerVation Protocol. IEEE Network, 7(9), 1993.
4. J. Castro and N. Nabona. Nonlinear multicommodity network flows through primal partitioning and comparison with alternative methods. In Proc. of the 16th IFIP Conference, System Modeling and Optimization, 1994.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness.* W. H. Freeman and Company, 1979.
6. B. Gleeson, et. al. A Framework for IP Based Virtual Private Networks. Internet RFC 2764, 2000.
7. R. Guerin, A. Orda and D. Williams. QoS routing mechanisms and OSPF extensions. In *Proc. 2nd Global Internet Miniconference*, 1997.
8. M. Kodialam and T. V. Lakshman. Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *Proc. of IEEE Infocom,* 2000.
9. T. V. Lakshman and D. Stidialis. High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching. *Proc. of ACM Sigcomm*, 1998.
10. E. Rosen, A. Vishwanathan and R. Callon. Multi-Protocol Label Switching Architecture. Internet RFC 3031, 2001.
11. V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast Scalable Level Four Switching. *Proc. of Sigcomm*, 1998.
12. V. Srinivasan, S. Suri and G. Varghese. Packet Classification using Tuple Space Search. *Proc. of Sigcomm*, 1999.

Towards Better Support of Transaction Oriented Communication in Differentiated Services Networks

Roland Bless, Dirk Holzhausen, and Klaus Wehrle

Institute of Telematics, Universität Karlsruhe (TH),
Zirkel 2, D-76128 Karlsruhe, Germany
Phone: +49 721 608 6411, Fax: +49 721 388097
{bless,holzhaus,wehrle}@tm.uka.de

Abstract. Applications using transaction oriented communication have special requirements. Packet forwarding of a client's request and a server's response should be as fast as possible and also highly reliable. This implies employment of a communication service supplying minimal end-to-end delay and minimal packet loss. Currently defined per-hop forwarding behaviors (PHBs) and per-domain behaviors (PDBs) within Differentiated Services networks do not support these requirements of bursty traffic well enough. This paper proposes a new PHB and a suitable PDB called *"Quick Forwarding"* to fulfill the previously described requirements. Results of simulations are presented for evaluation of the newly proposed behavior and show that it performs better than other behaviors based on Expedited Forwarding or Assured Forwarding. This is a first step towards better support for this important class of applications.

1 Introduction

There are many applications that would profit from using services that are based on Differentiated Services (DS) [3,1] mechanisms. For instance, Internet telephony applications, Virtual Private Networks or applications that transmit continuous streams requiring a low and bounded jitter will preferably use Per-Domain Behaviors (PDBs, cf. [8]), such as the "Virtual Wire" PDB [7], based on the Expedited Forwarding (EF) Per-Hop Behavior (PHB) [6]. More elastic applications may use PDBs (e. g., "Assured Rate" [9]) based on the proposed Assured Forwarding (AF) PHB classes [2]. While they get an assured minimal rate, additional resources may be used if available. The latter are usually shared (at least) between all users of the same AF PHB class, therefore allowing some degree of statistical multiplexing. But packets using these additional resources (i. e., they exceed a committed burst size [9]) are more likely to be discarded during congestion situations.

However, an important class of applications is currently not supported "well enough" by the so far proposed PDBs and PHBs. Such applications show a certain typical property: a client sends a query message consisting of one or few more packets to a server which sends a (normally larger) reply message back to the client. Therefore, the resulting traffic shows a bursty behavior: a small series of packets is consecutively sent at peak rate (usually at link rate) to the peer (client or server respectively) after which usually a "silence period" follows during which no packet transmission occurs. In this paper, a message exchange with such properties is called *transaction oriented communication*, because of the query/reply (and possibly commit) exchange which is typical for database or directory queries and operations as well as for many signaling protocols, remote procedure calls (RPCs) or middleware infrastructures such as CORBA.

The requirements of this application class related to network communication services are mainly the following two: messages (that may consist of one or more packets) should be transmitted as fast as possible (all packets should possess a minimal end-to-end delay) and highly reliable. The first property is desirable since it shortens the overall transaction time which is very important for fast operations (e. g., RPCs). The second property serves the same purpose: if a packet gets lost it has to be retransmitted which adds additional delay. Currently proposed PDBs and PHBs do not support this application class "well enough" in the sense that transactions will last longer than necessary due to violation of one or both properties above. This is due to the fact that packets experience additional delay or have a higher drop probability than required for fast transactions.

Basically, Differentiated Services networks can constitute a good basis for enabling support for those applications. But existing per-hop behaviors, which principally determine how packets are treated while they are forwarded by a DS node, do not fulfill both properties very well.

The EF PHB [5,6] is a forwarding behavior for enabling "low delay, low jitter and low loss services". In order to achieve these properties EF queues should ideally contain no packets at most times (especially to accomplish low and bounded jitter). Thus, an incoming EF packet should leave a DS node in principle directly after its arrival, i. e., it sees no packets in the respective EF queue. Empty queues can be reached by ensuring that "the service rate of EF packets on a given output interface exceeds their arrival rate at that interface over long and short time intervals, independent of the load of other (non-EF) traffic" [5]. In order to restrict jitter, bursts of EF packets arriving at a DS node are not desirable for this PHB. Eliminating or reducing packet bursts usually requires traffic shaping at least at DS domain ingresses [7]. This however imposes an additional delay for a burst of packets that form a transactional query or reply. For fast transaction processing it is desirable that all packets of such a transactional message are forwarded and arrive *en bloc*, i. e., as a consecutive sequence. Consequently, the bursty nature of transaction oriented communication contradicts with the objective of having nearly empty EF queues.

In constrast to EF, the proposed AF PHB [2] allows bursts, but normally assigns a higher drop precedence to packets in bursts (it actually depends on the concrete PDB, but it will usually be the case when the committed information rate and burst size is exceeded [9]). An AF PHB class comprises a group of three AF PHBs that have at least two different drop precedences. The objective of assured forwarding is that transfers get at least their committed information rate, but when there are additional resources available, they can be used for faster transmission. Packets using those additional resources are marked with an AF PHB of higher drop precedence (e. g., allowing TCP to reduce its sending rate and to adapt to the available shared bandwidth). This would increase the drop probability of a burst's packets thereby violating the second property ("nearly no loss") from above.

In this paper a new behavior is described that yields a better support for the previously described transaction oriented communication. Especially, time critical applications such as banking and brokerage applications or real-time process control applications will profit from this better support.

The paper is organized as follows: the properties of a new forwarding behavior are defined in the following section 2. Section 3 describes an evaluation by simulation of the proposed behavior. The paper closes with concluding remarks and an outlook on future work in section 4.

2 A Quick Forwarding Behavior

We define a *Quick Forwarding* behavior that consists of a PHB and an appropriate PDB, i. e., (among other things) appropriate conditioning functions and assured quality of service attributes. Since the quick forwarding behavior should be "burst friendly" and should also provide a fast forwarding of packets as well as a low loss guarantee, the following conditions apply:

– The QF PHB uses own resources, i. e., a separate queue. Packets of higher priority, e. g., EF packets should always be served before any QF packets, but any unused EF bandwidth will be used by QF.
– It assures transport of packets that are sent with a mean rate, actually defined by the mean rate r and a maximum tolerable burst size b that is sent at peak rate p (commonly link rate). Therefore, after transmission of a burst, the sender must pause sending packets for a while.
– A DS node should forward all packets of a single burst as fast as possible, i. e., with maximum link output rate. Usually bursts will cumulate to even larger bursts in (output) queues. Thus, in order to empty the queue fast enough, the ratio of the quick forwarding behavior aggregate bandwidth to overall link bandwidth should be small.
– The queue should not grow over longer periods, since this would inevitably lead to packet loss. Thus, the queue should be emptied fast enough. Hence, the effective departure rate of a QF aggregate must be greater than its maximum arrival rate, i. e., the total rate of all incoming QF aggregates flowing into that aggregate. This condition has to be verified by admission control with respect to the given mean rates.

The suitable traffic conditioning actions [3] are as follows:

– Classifying and initial marking of traffic at the first boundary node (first-hop router) of a DS domain.

- Metering in the first-hop router by using a token bucket that has fill rate r and size $B := b(1 - r/p)$.
- Unconditional dropping of non-conforming packets (i. e., when the token bucket contains no or too few tokens).
- No traffic shaping at the domain ingress, since it would add additional delay to packets of a burst.

In addition to that, some resource management aspects have to be considered. *Admission control* has to be performed (e. g., by some DS management components such as Bandwidth Brokers) in order to guarantee transmission of data at the specified mean rate and to prevent packet loss. Both mean rate and maximum burst size have to be considered in admission control tests in order to avoid buffer overflows. Due to the fact that all injected bursts may be cumulated in the worst case, queues with large capacities have to be provided. Since the overall amount of traffic in the QF aggregate is limited by admission control and link rates should be much higher than the configured QF rate, those queues are also emptied quickly.

At a first glance it may be non-intuitive why QF could forward packets quicker than EF, because EF packets should very rarely see any packets before them in their output queue. But one important property of EF is to restrict jitter, thus bursts or filled queues are not desirable by definition [5,6]. QF avoids any traffic shaping and always serves output queues at link rate, thereby allowing a faster forwarding of packets as illustrated and confirmed in the following section.

3 Evaluation

To show that using QF is indeed faster than using EF for the considered transaction oriented communication resp. bursty traffic, several simulations were performed. A modular QoS simulation suite called simulatedKIDS [13] was used for this purpose. It is a very flexible construction set of basic building blocks for QoS mechanisms that can be combined to build nearly arbitrary QoS behavior. SimulatedKIDS is based on the freely available discrete event simulation system OMNeT++ [11]. Various scenarios with different traffic generators and topologies have been examined, but due to space limitations, only two of these scenarios are described here.

The implementation of QF was evaluated using two different variants of scheduling algorithms: one with simple priority scheduling (cf. Fig. 1) and one with a modified weighted fair queueing (WFQ) combined with a simple priority scheduling (to let EF always have priority). The latter implementation could only show significant better results than EF if there was unused bandwidth of any other PHBs (including EF) left. This is due to the fact that WFQ limits the bandwidth that QF can use, thus bursts may be delayed until QF gets its time slice again to serve queued packets. For the rest of the paper only the implementation based on simple priority scheduling is used.

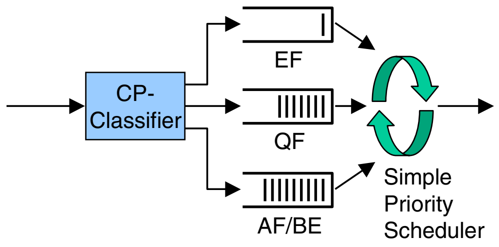


Fig. 1. DS mechanisms on an output interface of an interior router

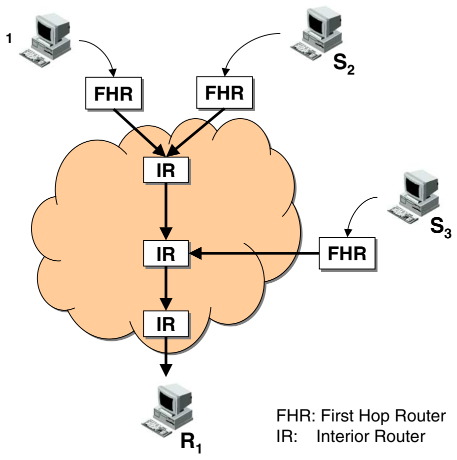


Fig. 2. A small topology

3.1 Scenario 1

The small DS domain that is depicted in Fig. 2 was used for validation of the implementation in a first simple scenario. Due to its low loss property and expedited packet forwarding at highest priority, setting up a "Virtual Wire" path [7] for transmitting important transaction oriented data would have been an alternative to using QF. Thus, a first comparison between EF and QF was done. Bursty traffic was generated by simple On/Off sources at each of the senders S_1, S_2 and S_3 and was sent towards the receiver R_1. The traffic mean rate was set to 10 Mbit/s and the burst length to 40 kbyte. All packets had a fixed length of 1250 byte. The first-hop routers (FHR) had a link rate of 100 Mbit/s and interior routers (IR) a link rate of 150 Mbit/s. All queues were large enough so that no packet losses occured due to completely filled queues. The configured share of EF and QF were each 20 Mbit/s at FHRs and 60 Mbit/s at IRs, thus the bandwidth shares of EF and QF were identical. Links were additionally filled up

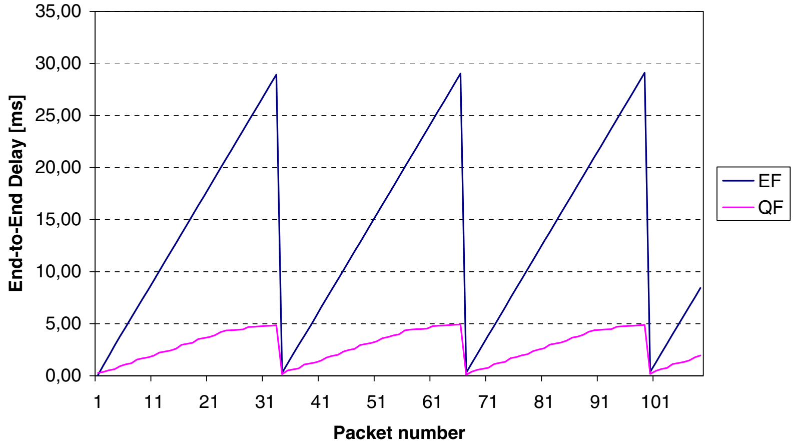


Fig. 3. End-to-End delay of packets received at R_1 (minimum: 0.067 ms)

(saturated) by best-effort traffic. Loss of QF or EF packets did not occur during the simulation.

The advantage of using QF instead of EF for such bursty traffic is illustrated in Fig. 3. Traffic generators are sending a burst of packets at link rate ("On" period) and then pause for a while to comply with the configured mean rate ("Off" period). This results in the depicted sawtooth curves, because a packet within a burst has to wait in the queue until all preceding packets of the burst have been transmitted. Traffic shapers release EF packets at the configured rate after the first router, resulting in a continuous data flow without bursts. But due to the artifical delay inserted by the traffic shaper, a stronger increasing end-to-end delay for adjacent EF packets during a burst can be observed as expected.

3.2 Scenario 2

The second scenario presented here is more complex and is illustrated in Fig. 4. In addition to the previous scenario, an AF class (PHB group) with two different drop precedence levels is also simulated. The default PHB for best-effort traffic is mapped to the AF PHB with highest drop precedence AF13, while packets with medium and low drop precedence are treated identically. Therefore, AF and BE share the same queue as depicted in Fig. 1. This configuration lets AF excess traffic share the residual bandwidth with BE traffic (cf. first example in (b) of section 7 in [2]).

There are 9 access IP subnets that are connected via first-hop routers to the DS domain. The FHRs have a link bandwidth of 100 Mbit/s while interior routers have links up to 300 Mbit/s. For every EF, QF and AF PHB, self-similar and bursty traffic is generated by 4 independent Pareto distributed On/Off-sources (with cut-off). The burst size varied between 30 and 75 kbyte while the off-time

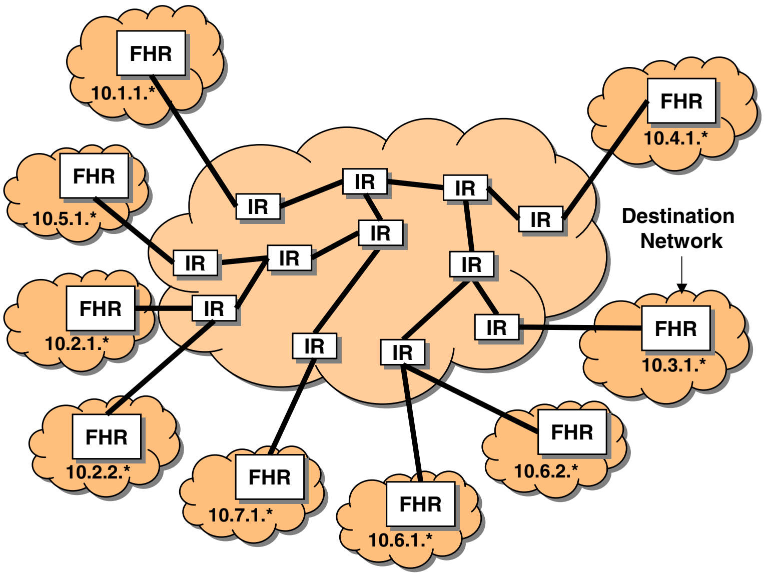


Fig. 4. A large DS domain, used for evaluation in scenario 2

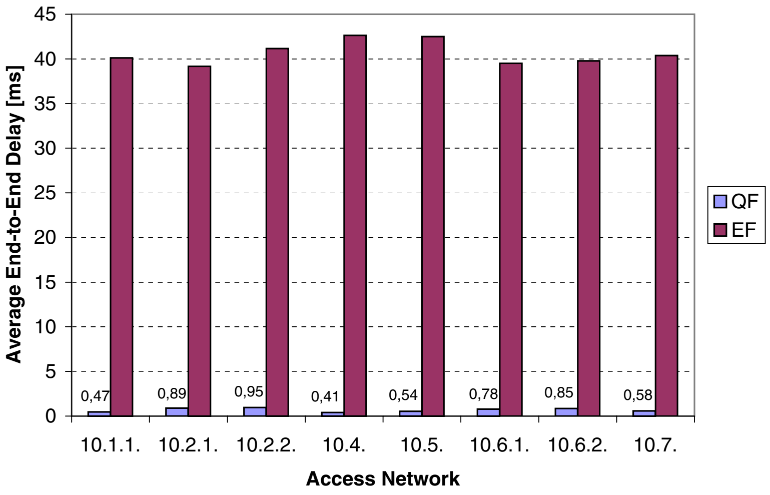


Fig. 5. Average end-to-end delay (in ms) of QF and EF packets for every access network to destination network 10.3.1.0/24

varied between 0.25 s and 5 s. Thus, taking all 4 traffic generators together, the mean rate for every PHB was roughly about 10 Mbit/s. Best-effort traffic was sent at link rate so that it used up any residual bandwidth. All traffic was sent to the access network 10.3.1.0/24. Several simulation runs were executed with a duration of 500 s simulated time each.

Again, loss of QF or EF packets did not occur during the simulation. The maximum required buffer size for the QF queue in the last interior router before the destination network was 2.4 Mbyte to prevent packet loss. Figure 5 shows a comparison of average end-to-end delays between QF and EF for each subnet. In this scenario, QF packets are transmitted at least one order of magnitude faster than EF packets. This is mainly due to the traffic shaping of EF packets to an aggregate rate of 10 Mbit/s at all domain ingress points (FHRs), whereas QF packets can be forwarded with 100 Mbit/s.

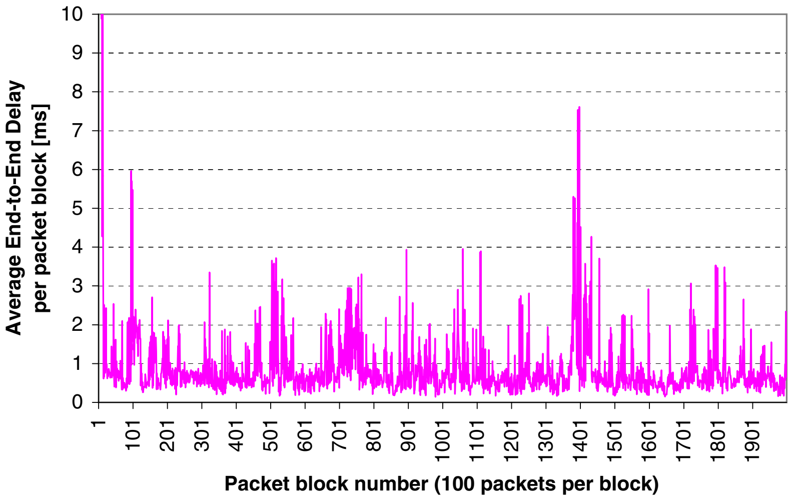


Fig. 6. End-to-end delay (in ms) averaged over a packet block of 100 QF packets from network 10.2.2.0/24

A more detailed view of the achieved end-to-end delay for QF packets is presented in Figure 6 where end-to-end delay of packets from network 10.2.2.0/24 is averaged over an interval of 100 packets (packet block, numbered on x-axis). Although there are several peaks present due to aggregation effects, the overall average end-to-end delay value lies around 1 ms.

The difference in order of several magnitudes between end-to-end delay values of QF, EF and also AF packets can easily be seen in the Figure 7 that uses a logarithmic scale on the y-axis. Delay of AF packets is even worse, since they share one queue with best-effort packets in this particular implementation.

The current delay jitter of packets (calculated by subtracting end-to-end delay values of consecutive packets) was also examined. Additionally, the delay jitter behavior of QF is not worse than EF, it is even better as Figures 8(a) and 8(b) show. Please note that the scale in Fig. 8(a) ranges from -140 ms to 80 ms whereas only from -4 ms to 3 ms in Fig. 8(b). This is due to the fact that bursts of EF packets are stretched by traffic shapers at the domain ingress, thus enlarging the individual delay jitter of packets.

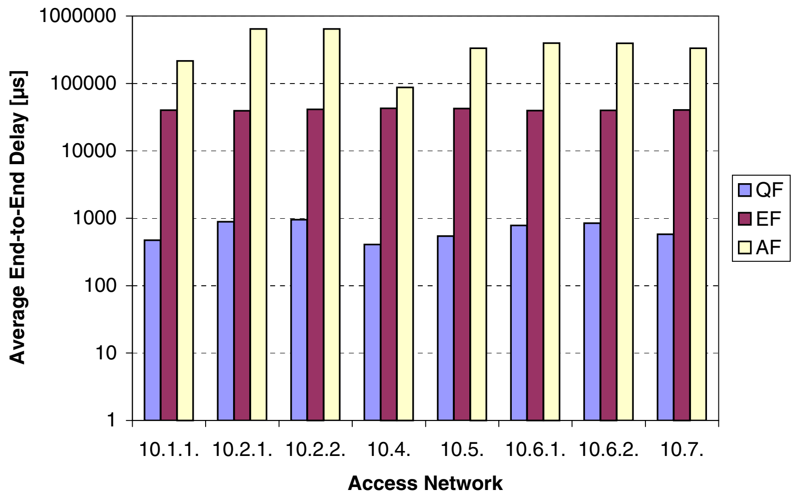
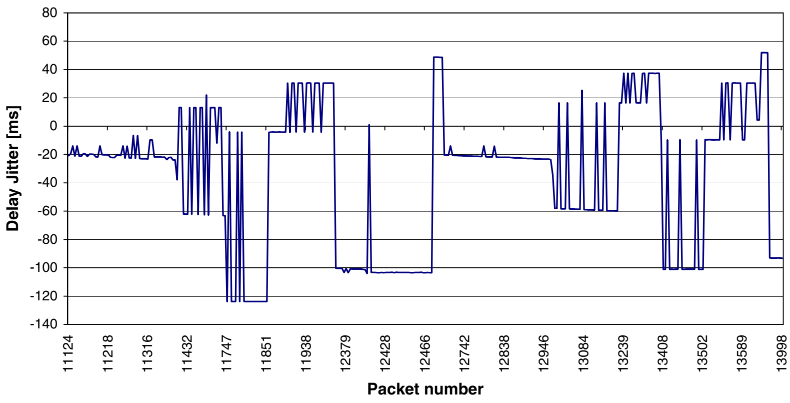


Fig. 7. Average end-to-end delay (in μs) of QF, EF and AF packets for every access network (logarithmic scale on y-axis)

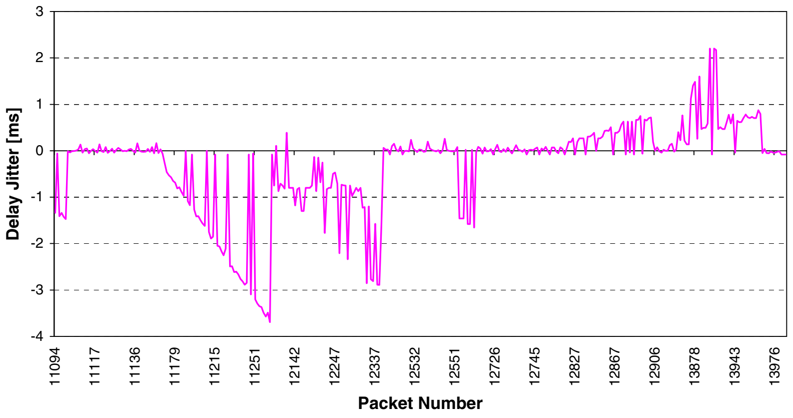
4 Conclusion and Outlook

The presented simulations have shown that the defined Quick Forwarding behavior is able to support transaction oriented communication better than other currently existing behaviors. Particularly, the EF PHB was chosen for comparison, because up to now it would have been an alternative to set up "virtual leased lines" [7] for transmission of transaction oriented traffic to use the low end-to-end delay and low loss properties of EF. Now, by using Quick Forwarding a better possibility exists. Many client/server applications or every kind of control and management traffic will profit from a service based on Quick Forwarding. Even when TCP is used as a reliable transport protocol, bursts of several segments (up to current maximum window size) will typically be generated if the window is completely opened once. Because QF offers also a rate guarantee, TCP connections that use QF will not experience congestion. Consequently, there is always an open window comprising several segments. However, for transaction oriented applications a better suited protocol such as SCTP [10] (it preserves message boundaries) should be used if available.

On the one hand, there are large buffers needed in order to prevent packet loss. On the other hand, providing large buffers would be better than having to retransmit data resulting in a much larger end-to-end delay. Large buffers that potentially hold many packets are not a problem for QF whereas it is for EF due to the requirement of almost empty queues. Since a QF queue is always served at link speed, high bandwidth links will empty those buffers faster. The actual gain of QF is caused by avoiding traffic shaping (and using WFQ for QF which has a similar limiting effect) at domain ingresses. Thus, QF packets are always forwarded with the highest possible rate. However, damage must be limited that



(a) EF packets



(b) QF packets

Fig. 8. Delay Jitter in *ms* of some packets from network 10.2.2.0/24

QF traffic could inflict on other traffic, because forwarding of other lower priority classes' packets is suspended until the QF queue is emptied. Therefore, the ratio of QF aggregate bandwidth to overall bandwidth should be not too large. The latter condition must be enforced by using admission and usage control, thus effectively controlling the overall amount of QF traffic (indirectly limiting QF flows and users, too). In our simulations the QF share was set to 20% of the overall bandwidth. Note that unused QF bandwidth can be used by other PHBs of lower priority.

However, it may be difficult to exactly characterize and predict application behavior in advance. Specifying a mean rate may not be flexible enough for all applications. Nevertheless, it allows some kind of admisson control which is essential for providing the desired guarantees.

Our simulations prove that providing a QF within a DS domain (i. e., a reasonable PDB) is feasible, but more experience and analysis is needed for providing an end-to-end service. This is, however, also true for other currently defined PDBs. Large buffer sizes for QF queues in order to prevent packet losses may be required, but can be effectively controlled by the admission control procedure. To reduce the required sizes it may be possible to use some degree of statistical multiplexing thereby reducing the loss guarantee to quantifiable percentiles. In the worst case, traffic shaping for QF aggregates will be occasionally required for long end-to-end paths. Therefore, we will investigate possibilities for a new kind of (to some degree) "burst-friendly" re-shaping mechanisms.

Future work will include extended simulations of several DS domains as well as an evaluation by a real implementation using an updated version of our Linux-based DS implementation architecture KIDS [12,4]. Determining a sensible and suitable QF share is of particular interest during our further research.

References

1. F. Baker, D. Black, S. Blake, and K. Nichols. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, Dec. 1998.
2. F. Baker, J. Heinanen, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC2597, June 1999.
3. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. RFC 2475, Dec. 1998.
4. R. Bless and K. Wehrle. Evaluation of Differentiated Services using an Implementation under Linux. In *Proceedings of the 7th IFIP Workshop on Quality of Service, London, June 1999*. IEEE, 1999.
5. B. Davie, A. Charny, F. Baker, J. Bennet, J.-Y. Le Boudec, A. Chiu, W. Courtney, S. Davari, V. Firoiu, C. Kalmanek, K. Ramakrishnam, and D. Stiliadis. An Expedited Forwarding PHB. Internet draft – draft-ietf-diffserv-rfc2598bis-01.txt, Apr. 2001.
6. V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. RFC 2598, June 1999.
7. V. Jacobson, K. Nichols, and K. Poduri. The 'Virtual Wire' Per-Domain Behavior. Internet draft – draft-ietf-diffserv-pdb-vw-00, July 2000. Work in progress.
8. K. Nichols and B. Carpenter. Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification. RFC 3086, Apr. 2001.
9. N. Seddigh, B. Nandy, and J. Heinanen. An Assured Rate Per-Domain Behaviour for Differentiated Services. Internet draft – draft-ietf-diffserv-pdb-ar-00, Feb. 2001. Work in progress.
10. R. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream control transmission protocol. RFC 2960, Oct. 2000.
11. A. Varga. Omnet++ discrete event simulation system 2.0. http://www.hit.bme.hu/phd/vargaa/omnetpp.htm, Feb. 2000.
12. K. Wehrle. An Open Architecture for Evaluating Arbitrary Quality of Service Mechanisms in Software Routers. In P. Lorenz, editor, *Networking - ICN 2001*,

Lecture Notes of Computer Science 2094, pages 117–126. Springer, 2001. First International Conference on Networking (Proceedings Part II), Colmar, France, July 9-13, 2001.

13. K. Wehrle, V. Kahmann, and J. Reber. A simulation suite for the Internet Protocol with the ability to integrate arbitrary QoS behavior. Proceedings of the Computer Networks and Distributed Systems Modelling and Simulation Conference (CNDS'01), Phoenix, Jan. 2001.

The Olympic Service Model: Issues and Architecture

Albert Banchs[1,2], Olga Leon[1,2], and Sebastia Sallent[2]

[1] C&C Research Laboratories, NEC Europe Ltd., Heidelberg, Germany
[2] Departament de Telematica, Universitat Politecnica de Catalunya, Spain

Abstract. The Olympic Service Model is a proposal for providing service differentiation in the Internet. With this model, those users who pay more receive a higher amount of network resources, based on a three class granularity (bronze, silver, gold). However, the amount of resources received by a user is not specified, and depends on the level of congestion at a given time. In this paper we analyze the validity and limitations of the Olympic Service Model. We then propose an architecture, the SSD architecture, which provides service differentiation according to this model, both for the intra-domain and the inter-domain cases. Finally, we compare via simulation our approach with other existing architectures of the Olympic Service Model.

1 Introduction

The current Internet is built on the Best Effort model where all packets are treated as independent datagrams and are serviced on a FIFO basis. This model suffers fundamentally from two problems: the potentially unfair distribution of the network resources and the lack of differentiation.

The potentially unfair resource distribution problem results from the fact that the Best Effort model does not provide any form of traffic isolation inside the network and relies on the application's behavior to fairly share the network resources among the users. Therefore the cooperation of the end systems (such as provided by TCP congestion control mechanisms) is vital to make the system work. In today's Internet, however, such dependence on the end systems' cooperation for resource distribution is increasingly becoming unrealistic. Given the current Best Effort model with FIFO queuing inside, it is relatively easy for non-adaptive sources to gain greater shares of network bandwidth and thereby starve other, well-behaved, TCP sources. For example, a greedy source may simply continue to send at the same rate while other TCP sources back off. Today, even many applications such as web browsers take advantage of the Best Effort model by opening up multiple connections to web servers in an effort to grab as much bandwidth as possible.

The lack of differentiation relates to the incapacity of the Best Effort model to provide a better service to those consumers who are willing to pay more for it. In today's Internet there is a growing demand for user differentiation based on their services' needs. For example, there are many companies relying on the

Internet for day-to-day management of their global enterprise. These companies are willing to pay a substantially higher price for the best possible service level from the Internet. At the same time, there are millions of users who want to pay as little as possible for more elementary services. Since the Best Effort model treats all packets equally (*same-service-to-all* paradigm), it does not allow Internet Service Providers (ISPs) to differentiate among users as needed.

Over the last ten years, considerable effort has been made to provide Quality of Service (QoS) in the Internet, leading to the specification of an Integrated Services architecture for the Internet (IntServ) [1]. However, research and experience have shown a number of difficulties in deploying the IntServ architecture, due to scalability and manageability problems. The scalability problems arise because IntServ requires routers to maintain control and forwarding state for all flows passing through them. Maintaining and processing per-flow state for gigabit and terabit links, with millions of simultaneously active flows, is significantly difficult from an implementation point of view. The manageability problems come from the lack of support for accounting, the high administrative overheads and the complexity of inter-ISP settlement.

The above issues have led to a number of proposals for providing *differentiated services* in the Internet. In those proposals, scalability is achieved by pushing most of the complexity and state to the network edges (where both the forwarding speed and the number of flows are smaller); at the edge, incoming packets are classified among several classes, and core routers do not need to store state for each flow, but can instead process packets using different policies for each traffic class. In a similar way, manageability is achieved by focusing on traffic aggregates instead of individual flows, where a traffic aggregate is a large set of flows with similar service requirements.

IETF's Differentiated Services (DiffServ) architecture [2,3,4] solves the potentially unfair resource distribution problem of the Best Effort model by performing some type of admission control at the edge of the network. Admission control ensures that no user sends more traffic than he/she is allowed to. A key point for admission control is to determine how much traffic a user should be allowed to send, such that the network does not become congested and, therefore, can give the service expected. The difficulty lies, however, in estimating at the edge the congestion level to which the acceptance of a certain amount of traffic would lead[1]. One possibility is to use a static over-provisioned configuration. In this case, since the admitted traffic is always much smaller than the network resources, the danger of congestion is minimized. A more dynamic solution is the use of bandwidth brokers (BB), which are agents responsible for allocating network resources among users. In this approach, the knowledge of the network usage is centralized at the BB and the admission decisions to be taken are transferred from this BB to the edge. The design and implementation of BB is an ongoing effort [5].

[1] Note that this problem does not arise in the Integrated Services architecture, since in that architecture, the admission control decision is taken individually at each router on the path between sender and receiver(s) based on the local state information.

The Olympic Service Model has been designed to solve the problems of the Best Effort model while avoiding the complexity of the admission control schemes proposed for IETF's DiffServ architecture. In this paper we propose an architecture based on the Olympic Service Model and we compare it with existing architectures based on the same model.

The rest of the paper is structured as follows: In Section 2, we present the Olympic Service Model and discuss its validity and limitations. In Section 3, an architecture based on this model is proposed: the Scalable Share Differentiation (SSD) architecture. Section 4 validates the SSD architecture through simulations, comparing it with the other existing approaches for the Olympic Service Model. Finally, Section 5 gives a summary and concludes the paper.

2 The Olympic Service Model

Research on DiffServ is proceeding along two different directions: those proposals that use admission control and those that do not.

In the approach with admission control, it is possible to control the amount of traffic allowed inside the network. In this way, traffic that would decrease the network service to a level lower than a desired limit can be stopped and an admitted user can be assured of his/her requested performance level. This approach, which we refer to as *Absolute Service Differentiation*, can be considered as trying to meet the same goals as IntServ, i.e. absolute performance levels, but pushing complexity admission control and traffic policing to the edge and to the BB and thus avoiding per-flow state in the core routers.

The second approach, which we refer to as *Relative Service Differentiation*, cannot prevent flooding of the network using admission control, and the only option to provide service differentiation is to forward packets in the network nodes with a quality according to their relative priority. Therefore, absolute performance levels are not guaranteed and only relative ones can be provided. The advantage of *Relative Service Differentiation* is that it is easier to deploy and manage.

One of the models proposed for the *Relative Service Differentiation* is the Olympic Service Model [3]. This model consists of three service classes: in order of increasing priority, *bronze*, *silver*, and *gold*. Packets are assigned to these classes according to the service contracted by their sender. Then, packets assigned to the gold class experience a better treatment than packets assigned to the silver class, and the same type of relationship exists between the silver and bronze classes. The Olympic Service Model must be strongly coupled with a pricing scheme that makes the gold class more costly than the silver class and the silver class more costly than the bronze class.

2.1 Olympic Service Model for Elastic and Real-Time Traffic

In the following, we justify the validity of the Olympic Service Model for Elastic Traffic. We will base the study on utility functions. Utility functions map network parameters into the performance of an application: they reflect how the performance of an application depends on the network parameters. With this definition, the Olympic Service model will be of utility to a user when an increase in the priority of the class contracted by a user reflects an increase in the utility experienced by this user.

In [6], applications are divided into two groups (elastic applications and real-time applications) and qualitative utility functions are given for each group. Examples of elastic applications are file transfer, electronic mail and remote terminal. These applications experience a diminishing marginal rate of performance enhancement as network resources are increased, so the function is strictly concave everywhere. This is illustrated in Figure 1. We can observe from this figure that even though elastic applications benefit from an increasing amount of resources, they can still work with a low amount of network resources.

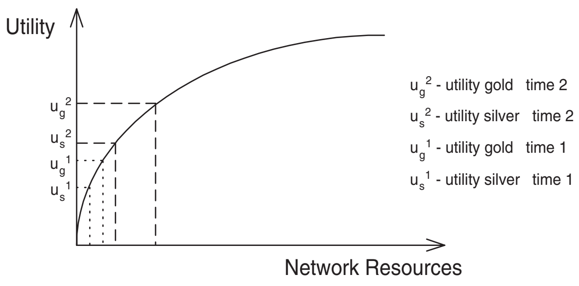


Fig. 1. Utility function for elastic applications

In the Olympic Service Model the amount of network resources received by each user is not defined but depends on the level of congestion in the network. This uncertainty about the amount of network resources associated to a class is not particularly harmful in the case of elastic applications since, as said above, this kind of applications can still work with a low amount of network resources. Also, with the Olympic Service Model, the higher the priority of the class contracted by a user, the more network resources this user will receive. As a consequence, a higher priority class will always lead to a higher utility for elastic traffic, independent of the level of congestion in the network:

$$u_i = f(congestion), \forall t, p_i > p_j \Rightarrow u_i > u_j \tag{1}$$

where p_i is the class priority (i.e. bronze, silver or gold) and u_i is the utility experienced.

Since with the Olympic Service Model, elastic applications always experience a positive performance, and this performance increases with the class priority, we conclude that this model provides a valid service for elastic traffic. Note that, with this approach, admission control can be avoided while still providing a good service for this type of traffic.

Real-time applications, in contrast to the elastic ones, need a minimum amount of network resources to perform well, and perform badly with an amount of resources lower than this threshold. Examples of such applications are link emulation, audio and video. The qualitative utility function for real-time applications is illustrated in Figure 2.

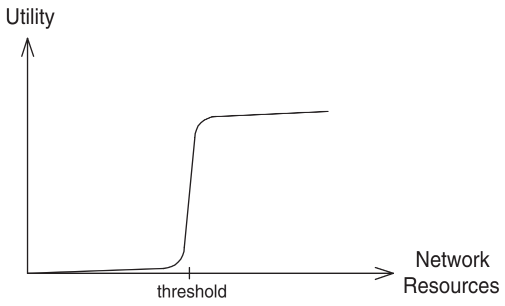


Fig. 2. Utility function for real-time applications

The uncertainty about the amount of network resources associated to a class in the Olympic Service Model may lead to a high priority class receiving a lower amount of resources than the minimum required and, consequently, experiencing a null performance. Also, with this model, an increase in the class priority will not always be beneficial: it will only be beneficial if it leads to an amount of network resources higher than the minimum required.

We conclude that the Olympic Service Model is not appropriate for real-time applications. This kind of applications would be better handled with admission control: without admission control, the level of congestion of the network cannot be controlled, and, as a consequence, utility cannot be guaranteed to applications that need a certain amount of resources to work properly. The Expedited Forwarding of IETF's DiffServ architecture [4] handles real-time traffic in this way. In [7] we provide a real-time traffic extension for the SSD architecture presented here. This extension relies on a user-based admission control scheme to guarantee that a real-time application receives the required amount of resources.

2.2 Sender-Based Approach

The fact that the Olympic Service Model is sender-based imposes some limitations to its functionality. With a sender-based approach, a user can influence the treatment experienced by the packets he/she is sending, but not the treatment experienced by the packets he/she is receiving.

The services that fit best the nature of a sender-based approach such as the Olympic Service Model are the *one-to-one* and *one-to-any services for sending*. An example of the one-to-one case could be a VPN service that connects two networks of a company; if the gold service class is contracted for both networks, the VPN service will experience a good quality. An example of the one-to-any case could be a company that does its business on the web and is willing to pay an additional price to provide its users with a fast feel of its web site. If this company contracts the gold service class, its users will experience a good service quality.

While the *one-to-one service for sending* can be relatively easily provided by IETF's DiffServ architecture, the *one-to-any service for sending* is much more complex and it is still an ongoing effort [8].

A *one-to-one service for receiving* does not match the nature of the Olympic Service Model, which is sender-based, but it still could be indirectly provided. An example of such a service could be a user that frequently accesses a specific video server to download movies [9]. With the Olympic Service Model, this user could contract a high quality service with the video server, which would in turn contract the gold service class with the network for the delivery of movies to this specific user. This would result in a good service quality experienced by the user when using this service. Note that in this case the money transaction consists of two steps: a first step from the user to the video server and a second step from the video server to the network operator.

Finally, the *any-to-one service for receiving* cannot be supported by the Olympic Service Model. An example of such a service could be a user willing to pay an extra price for high speed Internet access. Due to the usefulness of this service, the lack of support for it is an important limitation. The problem with any-to-one services is that they necessarily require some kind of signaling between the user and the ingress point of the packets received by him/her. Since the lack of signaling strongly contributes to the simplicity of an architecture, we conclude that the Olympic Service Model trades off functionality with simplicity. Note that within the IETF's DiffServ architecture, the support of any-to-one services has not been addressed yet.

3 Scalable Share Differentiation

In this section, we propose an architecture that implements the Olympic Service Model. In our architecture, each class of the Olympic Service Model (bronze, silver and gold) is mapped to a share, in such a way that the bandwidth experienced by a user is proportional to the share that he/she has contracted. The share associated with each class is chosen by the network operator and depends on the desired level of differentiation between classes; however, the gold class should be mapped to a higher share than the silver, and the silver to a higher than the bronze. This is expressed by the following equation:

$$\frac{r_g}{S_g} = \frac{r_s}{S_s} = \frac{r_b}{S_b} \tag{2}$$

where r_g is the bandwidth that a user would experience if he/she contracted the gold service class, r_s the bandwidth that he/she would experience with the silver class and r_b with the bronze, and S_g, S_s and S_b are the shares associated with each class ($S_g > S_s > S_b$).

The architecture proposed does not need to keep per-user state in the core nodes and therefore scales well with the number of users. We have called this architecture SSD (*Scalable Share Differentiation*). The SSD architecture consists of two parts: intra-domain and inter-domain.

3.1 Intra-domain

In the SSD architecture, each user i contracts a service class with the network operator, and this service class is mapped to a share S_i (where $S_i = S_g$, S_s or S_b). S_i is used to determine the treatment of the users' packets in bottleneck nodes.

The share of a user is divided equally among his/her packets in such a way that each packet gets assigned a weight $W_i = S_i/r_i$, where r_i is the total rate at which the user is transmitting. The share assigned to a user can be understood as some "wealth" that has been given to this user by the network operator. Following this analogy, when the user assigns weights he/she is distributing the amount of "wealth" assigned to him/her among his/her packets. Therefore, the weight of a packet represents its associated "wealth". Obviously, the proposed scheme has to assure that the more "wealth" is associated to a packet, the better treatment it should receive from the network.

To assign weights to the packets, the transmission rate of each user r_i is measured at the ingress node, and the corresponding value of W_i is inserted into the packet according to the concept of *dynamic packet state (DPS)* [10]. The basic idea of DPS is that each packet header carries some additional state information, in this case the weight W_i of the packets' originator, which is initialized by the ingress node of the diffserv network and processed by the core nodes on the path. Interior nodes use this information to possibly update their internal states and to update the information in the header before forwarding the packet to the next hop.

Within the SSD architecture, the specific DPS mechanism of inserting W_i into the packets is used to provide relative share differentiation: interior nodes use the packets' weight to determine the dropping policy for that user's packets in congestion situations. Whenever there is not enough bandwidth for that traffic type to serve all incoming packets, the packets with a lower weight should be dropped with higher probability.

That means, when a packet has too low a weight, its user distributed its share among too many packets (i.e., the user sent at a higher rate than allowed according to the user's contracted bandwidth share). In this case, the router drops some of the packets of that user, thereby reducing his/her packet rate at this link. Then, the node redistributes the share of the user among the fewer packets, which leads to a higher value of W_i for the remaining packets of that user.

Therefore, there is a certain value of weight, below which packets must be dropped with a certain probability to dissolve the congestion. We have called this value the *fair weight* W_{fair}. The probability of dropping an incoming packet with a weight lower than W_{fair} will be calculated in such a manner that the packets that are not dropped get assigned a weight that equals W_{fair}.

Let us define d_i to be the probability of dropping a packet belonging to user i at some node. Then the redistribution of the user's share among the packets that are not dropped leads to:

$$W_i^{new} = \frac{W_i^{old}}{1 - d_i} \tag{3}$$

Taking into account that packets with a weight lower than W_{fair} should adjust themselves to this value (i.e., their new weight should be equal to W_{fair}) and packets with a weight higher than W_{fair} can pass untouched, we have that, given the *fair weight* W_{fair}, the dropping probability of the packets of user i can be calculated with the formula:

$$d_i = \begin{cases} 0 & W_i > W_{fair} \\ 1 - \frac{W_i}{W_{fair}} & W_i < W_{fair}, W_i^{new} = W_{fair} \end{cases} \tag{4}$$

The algorithm used to forward packets in a SSD node, resulting from the equations above, is illustrated in Figure 3.

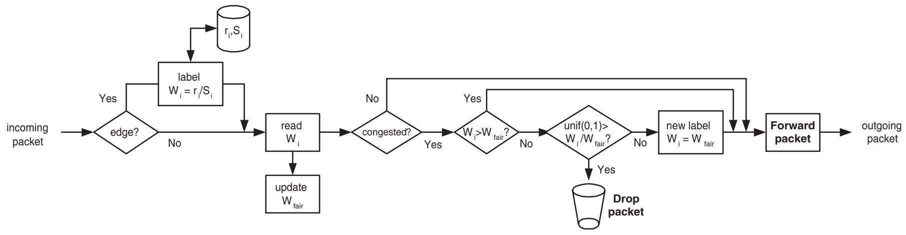


Fig. 3. Forwarding algorithm in SSD

With this algorithm, the rate experienced by flow j of user i is equal to:

$$r_{ij}^{out} = r_{ij}^{in}(1 - d_{ij}) = r_{ij}^{in} \frac{W_i}{W_{fair}^j} = \frac{r_{ij}^{in}}{r_i^{in} \cdot W_{fair}^j} S_i \tag{5}$$

where r_{ij}^{out} is the rate experience by flow j of user i, r_{ij}^{in} is the flow's sending rate, r_i^{in} is the total sending rate of user i and W_{fair}^j is the value of W_{fair} in the bottleneck link of flow j.

From Equation 5, the total rate experienced by user i, r_i^{out}, can be derived:

$$r_i^{out} = \sum_j r_{ij}^{out} = \left(\sum_j \frac{r_{ij}^{in}}{r_i^{in} \cdot W_{fair}^j} \right) S_i \tag{6}$$

From the above equation, it can be seen that the rate received by a user i will depend on the level of congestion of the links traversed by his/her flows, expressed by W^j_{fair}, but this rate will increase linearly with S_i. We conclude that the bandwidth experienced by a user is proportional to the share of the class contracted by this user (S_g, S_s or S_b), which is the objective that we stated for the SSD architecture in Equation 2.

A key point of the SSD architecture is the estimation of W_{fair} for each congested link. For scalability reasons, W_{fair} should be estimated without storing any per-user or per-flow information at the core nodes.

The problem of estimating W_{fair} is similar to the one solved by the CSFQ (*Core-Stateless Fair Queuing*) algorithm in [11]. In contrast to the SSD architecture, which focuses on the problem of distributing bandwidth among users, CSFQ focuses on bandwidth distribution among flows.

Both architectures differ not only in the problem they solve, but also in the way they solve it; for the estimation of W_{fair}, SSD applies small variations for every incoming packet, while CSFQ applies much bigger changes on a periodic basis. The advantage of the approach taken in SSD is that it adapts more quickly to network changing conditions (simulation results in [12] show that CSFQ tends to produce large errors under sudden changes from uncongested to congested). The solution adopted within the SSD architecture for the estimation of W_{fair} is presented in detail in [7].

3.2 Inter-domain

Since the Internet consists of different domains, in order to provide end-to-end QoS, domains are required to cooperate. Thus, the QoS behavior when crossing domains (inter-domain part) is an essential aspect of any DiffServ architecture. In the SSD architecture, the inter-domain part is, as the intra-domain, based on shares; but in this case it is not a user who contracts a share to his or her domain (where the share contracted by a user is a function of his/her service class, i.e. bronze, silver or gold), but it is a domain that contracts a share with a neighboring domain. This share that one domain contracts with another will be divided among all the users sending from the first domain to the second. So, for example, if a domain has 100 users sending to another domain, and wants them to experience the same quality as one user of the other domain with a share of 1 contracted within that domain, then the first domain will have to contract a share of 100 to the second domain.

When crossing domains, for scalability reasons users have to be somehow aggregated. As it has been explained in the intra-domain part of the architecture, per-user state needs to be maintained at the edges in order to measure each user's rate. If users are not aggregated, boundary routers also need to keep state for every user crossing the router. The number of users crossing edge routers will always be relatively small, but this does not have to hold true for boundary routers between domains. Therefore, if users are not aggregated, boundary routers may need to keep a very large state and will then become bottlenecks concerning scalability.

One possible way of aggregating users in our architecture would be to see all the users sending packets from one domain to another in the second domain as one user with a share of S_{d1} (where S_{d1} is the share that the first domain has contracted to the second). In this case all packets coming from the first domain would get assigned in the second domain a weight $W_{d1} = S_{d1}/r_{d1}$, where r_{d1} is the total sending rate from the first domain to the second. This solution, however, would not provide proper isolation; if there was a bottleneck in the second domain, all the packets coming from the first domain would be treated in the same way, independently of the share of their originator, and, as a consequence, the bandwidth assigned to each user would be independent of the service class contracted by him/her, violating thus one of the main goals of the Olympic Service Model. In order to provide proper isolation, when crossing domains the packets in the new domain should preserve the ratios between the weights they had in the old domain.

For that reason, the inter-domain architecture has to compute the weights for the packets coming from another domain in such a way that the ratios between the weights of the original domain are preserved and the overall effect in the new domain is the same as if a weight of $W_{d1} = S_{d1}/r_{d1}$ had been assigned to all incoming packets. The first condition is expressed in Equation 7. The second condition is equivalent to saying that the addition of the shares that the users from the first domain receive in the second domain should be equal to S_{d1}. This is expressed in Equation 8.

$$\frac{W_1^{new}}{W_1^{old}} = \frac{W_2^{new}}{W_2^{old}} = \ldots = \frac{W_n^{new}}{W_n^{old}} = \beta \tag{7}$$

$$S_{d1} = W_1^{new} \cdot r_1 + W_2^{new} \cdot r_2 + \ldots + W_n^{new} \cdot r_n \tag{8}$$

where W_i is the weight of the packets of user i and r_i the rate at which he/she is sending.

Combining Equations 7 and 8 leads to:

$$S_{d1} = \sum_i r_i \cdot \beta \cdot W_i^{old} \tag{9}$$

where β is the value we need to calculate in order to solve the problem (i.e., to obtain the new weights).

β could be obtained from Equation 9, but this would require keeping per-user information (specifically, the rate r_i at which each user i is sending). We already indicated that this solution is undesirable for scalability reasons.

One way of avoiding per-user state is calculating the average weight $\bar{W}$ of all packets going from the old domain to the new one. Extending Equation 9, we have:

$$\begin{aligned} S_{d1} &= \textstyle\sum_i r_i \cdot \beta \cdot W_i^{old} = \beta \cdot \sum_i r_i \cdot W_i^{old} \\ &= \beta \cdot r_{d1} \cdot \textstyle\sum_i \frac{r_i}{r_{d1}} \cdot W_i^{old} = \beta \cdot r_{d1} \cdot \bar{W} \end{aligned} \tag{10}$$

where the last term of Equation 10, the average weight $\bar{W}$ of incoming packets, can be calculated without the need of keeping per-user state.

Combining Equation 10 with $S_{d1}/r_{d1} = W_{d1}$, we obtain a way of calculating β without keeping per-user state:

$$\beta = \frac{W_{d1}}{\bar{W}} \tag{11}$$

Therefore, when crossing domains, the packets will be marked with the following *new weights*:

$$W_i^{new} = \frac{S_{d1}}{r_{d1} \cdot \bar{W}} \cdot W_i^{old} \tag{12}$$

With this mechanism, the level of differentiation in the second domain is preserved without need of storing per-user state in the boundary routers, and the users coming from the first domain receive in total an amount of network resources equal to S_{d1}.

4 Comparison with Existing Approaches

In this section, we compare via simulation our architecture with the existing approaches also based on the Olympic Service Model. The purpose of these simulations is, rather than validating the applicability of the SSD architecture in the current Internet, to show the validity of its conceptual approach for isolation and differentiation as compared to the other Olympic Service Model architectures.

For this purpose, we simulated a simple scenario with three users sending through one bottleneck link of 10 Mbps: user 1 with a share of 3 (gold class), user 2 with a share of 2 (silver class) and user 3 with a share of 1 (bronze class). This scenario reflects the level of isolation and differentiation achieved by users belonging to different service classes.

We also simulated another scenario with a variable number of users for each service class: three users for the gold class, two for the silver class and one for the bronze class. This second scenario reflects the level of isolation between the different users of the same class, and the impact of the load in a class.

Finally, we simulated a scenario with a two-domain network topology with a bottleneck link of 10 Mbps in the second domain, in order to study the level of isolation and differentiation provided by the different architectures when crossing domains.

For the above scenarios we used both UDP CBR sources sending at 5 Mbps and endless TCP sources, hereafter referred as UDP and TCP respectively.

4.1 SSD

Tables 1 and 2 show the level of isolation and differentiation provided by the SSD architecture for both the one-user-per-class and several-users-per-class cases. Simulation results show that, when all flows are UDP, SSD provides the desired level of differentiation independent of the number of users per class.

However, when the packet drops of the SSD architecture interact with the congestion control of TCP, results deviate from the desired ones, specially for the case when TCP and UDP flows compete with each other for bandwidth (tests 2 and 5). We conclude from these results that SSD does not provide perfect isolation between UDP and TCP. However, it still provides a fairly good level of isolation, taking into account the different level of aggressiveness of TCP and UDP sources.

The TBA-TCP architecture [13] proposes a modification of the TCP window computation, adjusting it to the most limiting W_{fair} in the forward path. This architecture, originally designed to be used with CSFQ, could also be used in SSD to improve the performance of TCP.

Table 1. SSD: One user per class

		TEST 1		TEST 2		TEST 3	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	4625	UDP	4979	UDP	4920
2	2	TCP	3744	TCP	3083	UDP	3385
3	1	TCP	1542	UDP	1825	UDP	1687

Table 2. SSD: Several users per class

		TEST 4		TEST 5		TEST 6	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	2052	UDP	2540	UDP	2131
2	3	TCP	2098	UDP	2544	UDP	2120
3	3	TCP	1995	TCP	1276	UDP	2087
4	2	TCP	1502	UDP	1670	UDP	1435
5	2	TCP	1493	TCP	767	UDP	1423
6	1	TCP	758	UDP	843	UDP	699

To test the inter-domain part of the SSD architecture we simulated a scenario in which the users of Table 2 are sending their flows to a second domain with which the first domain has contracted a share of 5, and in this second domain they are competing with a user that has contracted a share of 1 with the second domain. In the simulation results of Table 3 we can see that the differentiation between the users of the first domain is preserved in the second domain, and in total these users get approximately the share that their domain has contracted with the second domain. These were the objectives of the inter-domain architecture explained in Section 3.2.

4.2 User Share Differentiation (USD)

The User Share Differentiation (USD) architecture [14] also maps each service class into a share. The share corresponding to each user is stored by the core

Table 3. SSD: Inter-domain

		TEST 7		TEST 8		TEST 9	
user	share	source	Kbps	source	Kbps	source	Kbps
1-d1	3	TCP	1536	UDP	2122	UDP	1726
2-d1	3	TCP	1632	UDP	2086	UDP	1738
3-d1	3	TCP	1635	TCP	831	UDP	1747
4-d1	2	TCP	1050	UDP	1429	UDP	1162
5-d1	2	TCP	1106	TCP	576	UDP	1154
6-d1	1	TCP	546	UDP	677	UDP	594
d1	5	TCP	7507	TCP-UDP	7721	UDP	8121
7-d2	1	TCP	2253	UDP	1860	UDP	1660

nodes of the network. The core nodes use this share to give the packets of the corresponding user their fair part of the forwarding capacity. Note that since USD stores information for each user at core nodes, it has the problem of poorly scaling with respect to the number of users, and might result in implementation problems when applied to core routers in large domains.

The results for the intra-domain simulations (Tables 4 and 5) show that USD provides the required isolation and ensures the necessary differentiation according to the service classes. This perfect isolation can be achieved because in USD, in contrast to SSD, per-user state is stored at core routers.

For the inter-domain simulations of the USD architecture, we implemented user aggregation as suggested in [14], i.e., in the second domain, the users from the first domain are aggregated in classes with identical shares. In this case,

Table 4. USD: One user per class

		TEST 1		TEST 2		TEST 3	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	4999	UDP	4994	UDP	4996
2	2	TCP	3333	TCP	3336	UDP	3336
3	1	TCP	1666	UDP	1668	UDP	1667

Table 5. USD: Several users per class

		TEST 4		TEST 5		TEST 6	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	2142	UDP	2142	UDP	2142
2	3	TCP	2142	UDP	2142	UDP	2142
3	3	TCP	2142	TCP	2142	UDP	2142
4	2	TCP	1428	UDP	1428	UDP	1428
5	2	TCP	1428	TCP	1428	UDP	1428
6	1	TCP	714	UDP	714	UDP	714

users 1, 2 and 3 (gold class) are aggregated in a class in domain 2 with share 3, users 4 and 5 (silver class) in a class with share 2 and user 6 (bronze class) in a class with share 1. Note that since in USD users are statically assigned to a class, such a situation in which one class is more crowded than another can easily occur. The simulation results of Table 6 show that USD fails to guarantee proper differentiation due to user aggregation when crossing domains: all users receive the same treatment, independent of their service class. In addition, the inter-domain part of USD also fails to provide proper isolation, again due to aggregation effects: in test 8 we can see that TCP flows starve when sharing a class with UDP flows in the second domain.

Table 6. USD: Inter-domain

		TEST 7		TEST 8		TEST 9	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	1672	UDP	2485	UDP	1667
2	3	TCP	1672	UDP	2504	UDP	1669
3	3	TCP	1655	TCP	3	UDP	1662
4	2	TCP	1665	UDP	3248	UDP	1666
5	2	TCP	1667	TCP	48	UDP	1666
6	1	TCP	1666	UDP	1666	UDP	1666

4.3 SIMA

The Simple Integrated Media Access (SIMA) architecture [15,16] defines different levels of service based on the Nominal Bit Rate contracted by the user. SIMA provides two types of services, one for real-time traffic and the other for non-real-time. Since the focus of this paper is on elastic traffic, we will only consider the latter.

In SIMA, the sending rate of user i, r_i, is estimated at the ingress of the domain, and, based on this estimation, packets are labeled according to the following formula[2]:

$$L_i = \begin{cases} 6 & if x \geq 6 \\ int(x) & if 0 < x < 6 \\ 0 & if x \leq 0 \end{cases} \tag{13}$$

where x is

$$x = 4.5 - \frac{ln(r_i/NBR)}{ln(2)} \tag{14}$$

and $int(x)$ is the integer part of x.

Then, in core nodes, packets are dropped depending on their label L_i according to the Weighted Random Early Detection (WRED) active queue management algorithm with six separate thresholds, one for each label value.

[2] Note that since L_i can only take 6 different values, three bits are enough to store its value.

In the simulations of SIMA, we assigned a NBR of 3 Mbps to the users of the gold service class, 2 Mbps to the users of the silver service class and 1 Mbps to the bronze. Simulation results for the intra-domain case are shown in Tables 7 and 8. These results show that SIMA provides a good level of differentiation in the case of responsive TCP sources (tests 1 and 4), even though the throughputs obtained are not proportional to the NBRs. When dealing with non-responsive UDP sources, however, this feature is lost. In tests 2, 3, 5 and 6, we can see that the bronze service class starves when competing with the gold and silver service classes. This is due to the fact that packet drops in SIMA are not probabilistic but based on thresholds: a user sending too much, thus, will see all his/her packets dropped. Also in tests 2, 3, 5, and 6, the UDP silver sources receive the same treatment as the gold ones. This is due to the coarse granularity of SIMA, which comes from the small number of label values used: a user sending at 5 Mbps receives the same label both for a NBR of 2 and 3 Mbps, since

$$int\left(4.5 - \frac{ln(5/2)}{ln(2)}\right) = int\left(4.5 - \frac{ln(5/3)}{ln(2)}\right) = 3 \tag{15}$$

Finally, in test 5 it can be observed that the level of isolation provided by SIMA between UDP and TCP is not perfect but reasonable.

For the inter-domain simulations, we simulated a scenario in which the users of Table 8 are sending their flows to a second domain with which the first domain has contracted a NBR of 5 Mbps, and in this second domain they are competing with a user that has a NBR of 1 Mbps. In the simulation results of Table 9 we can see that SIMA does not provide proper differentiation when crossing domains: in tests 7 and 9 all users receive the same treatment, independent of

Table 7. SIMA: One user per class

		TEST 1		TEST 2		TEST 3	
user	NBR	source	Kbps	source	Kbps	source	Kbps
1	3 Mbps	TCP	5325	UDP	4977	UDP	4944
2	2 Mbps	TCP	3063	TCP	4721	UDP	4922
3	1 Mbps	TCP	1572	UDP	287	UDP	132

Table 8. SIMA: Several users per class

		TEST 4		TEST 5		TEST 6	
user	NBR	source	Kbps	source	Kbps	source	Kbps
1	3 Mbps	TCP	2330	UDP	2579	UDP	1991
2	3 Mbps	TCP	2331	UDP	2561	UDP	1978
3	3 Mbps	TCP	2357	TCP	1206	UDP	1985
4	2 Mbps	TCP	1140	UDP	2567	UDP	2015
5	2 Mbps	TCP	1128	TCP	1076	UDP	1995
6	1 Mbps	TCP	693	UDP	0	UDP	34

Table 9. SIMA: Inter-domain

		TEST 7		TEST 8		TEST 9	
user	NBR	source	Kbps	source	Kbps	source	Kbps
1-d1	3 Mbps	TCP	1420	UDP	1991	UDP	821
2-d1	3 Mbps	TCP	1596	UDP	2014	UDP	848
3-d1	3 Mbps	TCP	2023	TCP	1	UDP	819
4-d1	2 Mbps	TCP	818	UDP	1995	UDP	837
5-d1	2 Mbps	TCP	1622	TCP	0	UDP	826
6-d1	1 Mbps	TCP	1238	UDP	2006	UDP	834
d1	5 Mbps	TCP	8717	TCP-UDP	8007	UDP	4985
7-d2	1 Mbps	TCP	1210	UDP	1991	UDP	5011

their NBR in the first domain. In test 8 it can be seen that the inter-domain part of SIMA does not provide proper isolation either, since the TCP flows starve. This inter-domain behavior of SIMA is caused by the fact that the users from the first domain are treated like an aggregate in the second domain.

4.4 Delay Differentiation (DD)

In [17,18,19] different schedulers for proportional differentiation in delay are proposed. These schedulers basically schedule packets in such a way that the waiting time in the queue is inversely proportional to the share assigned to the corresponding service class.

In order to better understand the performance of this type of schedulers we ran the simulations corresponding to Tables 10 and 11 using the scheduler implementation of [20]. We can observe from the simulation results that delay differentiation provides a good level of differentiation for TCP traffic alone (tests 1 and 4), since the throughput obtained by a TCP flow is inversely proportional to its round-trip delay. For UDP traffic alone (tests 3 and 6), no differentiation in throughput is obtained; however, for this kind of traffic the delay alone may suffice to provide meaningful differentiation. The main drawback of the delay differentiation approach, however, is expressed by the results of tests 2 and 5. We can observe in these tests that TCP sources almost starve when competing with UDP, which shows that the queuing delay as differentiation parameter cannot provide proper isolation.

We conclude that, even though delay can be an important differentiation parameter for delay sensitive applications, it needs to be combined with bandwidth differentiation in order to provide proper isolation. This is the approach we have taken in the real-time extension of the SSD architecture that we have proposed in [7].

Inter-domain simulation results for this approach have not been provided since they do not differ from the intra-domain ones.

Table 10. DD: One user per class

		TEST 1		TEST 2		TEST 3	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	4884	UDP	4784	UDP	3266
2	2	TCP	3370	TCP	398	UDP	3360
3	1	TCP	1745	UDP	4842	UDP	3376

Table 11. DD: Several users per class

		TEST 4		TEST 5		TEST 6	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	2125	UDP	2481	UDP	1708
2	3	TCP	2132	UDP	2489	UDP	1643
3	3	TCP	2130	TCP	84	UDP	1676
4	2	TCP	1444	UDP	2455	UDP	1679
5	2	TCP	1444	TCP	20	UDP	1652
6	1	TCP	733	UDP	2469	UDP	1640

4.5 Class-Based Allocation (CBA)

A Class-based allocation approach, such as the one in [3], assigns a specific capacity to each service class. Each network flow that belongs to a certain class, therefore, shares a common set of resources with other flows in that class. This approach has the drawback that the service quality associated with a class is undefined, since it depends on the arriving load in that class: as the traffic in the Internet is extremely bursty [21], the load in each class and consequently its service quality fluctuates.

This behavior is shown in the simulation results of Tables 12 and 13. This simulations have been performed with the Weighted Round Robin implementation of [22], with a different queue for each class with a weight equal to the share of the class (3 for the gold class, 2 for the silver and 3 for the bronze). Table 12 shows that when the load in each class is uniform, the level of differentiation obtained is the desired. However, when the load in the higher priority classes is larger, the differentiation feature is lost: in the example of Table 13 we can see that all users get the same throughput, independent of the service class they have contracted (bronze, silver or gold). In addition, isolation is not provided inside each class, so the most aggressive sources eat up all the available bandwidth in a class (see test 5 in Table 13).

Inter-domain simulation results have not been provided for this approach either, since also in this case they do not differ from the intra-domain ones.

Table 12. CBA: One user per class

		TEST 1		TEST 2		TEST 3	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	4999	UDP	4994	UDP	4996
2	2	TCP	3333	TCP	3336	UDP	3336
3	1	TCP	1666	UDP	1668	UDP	1667

Table 13. CBA: Several users per class

		TEST 4		TEST 5		TEST 6	
user	share	source	Kbps	source	Kbps	source	Kbps
1	3	TCP	1666	UDP	2496	UDP	1666
2	3	TCP	1665	UDP	2498	UDP	1670
3	3	TCP	1664	TCP	2	UDP	1663
4	2	TCP	1665	UDP	3284	UDP	1670
5	2	TCP	1667	TCP	48	UDP	1662
6	1	TCP	1666	UDP	1666	UDP	1666

5 Conclusions

The Olympic Service Model is a pricing scheme that provides more network resources to those users who pay more. With this model users are not given absolute guarantees but relative ones: with the gold service class the quality experienced is better than with the silver service class, but this quality is left undefined and depends on the network conditions. Same kind of relationship exists between silver and bronze service classes.

The main advantage of the Olympic Service Model is its simplicity. The fact that the model is sender-based avoids the need of signaling, and its relative nature eliminates the need of admission control. However, this simplicity comes together with some limitations. With a sender-based approach, there are some services like Internet access that cannot be provided. The relative guarantees provided by the Olympic Service Model are well suited for elastic traffic, but not for real-time traffic, which requires of a more complex scheme with some kind of admission control providing absolute guarantees.

We conclude that the Olympic Service Model trades off functionality by simplicity, but still solves some major of the current Best Effort model, such as traffic isolation and service differentiation. Given the current difficulties in the deployment of IETF's DiffServ and IntServ models, mainly due to their complexity, a pricing scheme like the Olympic Service Model could find its application as the next step after the Best Effort model in the evolution of the Internet.

The SSD architecture implements the Olympic Service Model by allocating a fixed number of times more bandwidth to a user of the gold service class than to a user of the silver service class, and the same for the silver and bronze service classes. This level of differentiation between service classes is preserved when crossing domains.

The isolation and differentiation features of the SSD architecture have been validated via simulation. These features have been compared, also via simulation, with other architectures that also implement the Olympic Service Model, namely, User Share Differentiation (USD), Simple Integrated Media Access (SIMA), Delay Differentiation (DD) and Class-Based Allocation (CBA). Simulation results show that the SSD architecture is the only one that provides the isolation and differentiation features for both the intra-domain and the inter-domain cases.

References

1. R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998.
3. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.
4. V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," RFC 2598, June 1999.
5. "QBone Bandwidth Broker Advisory Council home page," http://www.merit.edu/working-groups/i2-qbone-bb.
6. S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1176–1188, September 1995.
7. A. Banchs and R. Denda, "A Scalable Share Differentiation Architecture for Elastic and Real-time Traffic," in *Proceedings of 8th International Workshop on Quality of Service IWQoS'2000*, Pittsburg, PA, June 2000.
8. M. Brunner, A. Banchs, S. Tartarelli, and H. Pan, "A one-to-any Probabilistic Assured Rate Per-Domain Behavior for Differentiated Services," Internet draft, February 2001, Available at http://www.icsi.berkeley.edu/~banchs/papers/one2any-AR-PDB.pdf.
9. C. Kalmanek, D. Shur, S. Sibal, C. Sreenan, and J. Merwe, "NED: A Network-Enabled Digital Video Recorder," IEEE LANMAN 2001, March 2001.
10. I. Stoica et. al., "Per Hop Behaviors Based on Dynamic Packet States," Internet Draft, draft-stoica-diffserv-dps.00.txt, Febuary 1999.
11. I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proc. ACM SIGCOMM 98*, Vancouver, Canada, August 1998, pp. 118–130.
12. Z. Cao, Z. Wand, and E. Zegura, "Rainbow Fair Queueing: Fair Banwdith Sharing Without Per-Flow State," in *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000.
13. R. Kapoor, C. Cassetti, and M. Gerla, "Core-Stateless Fair Bandwidth Allocation for TCP flows," in *Proceedings of IEEE ICC 2001*, Helsinki, Finland, June 2001.
14. Z. Wang, "User-Share Differentiation (USD): Scalable Bandwidth Allocation for Differentiated Services," in *HPN'98*, Vienna, 1998.
15. K. Kilkki, "Simple Integrated Media Access," draft-kalevi-simple-media-access-01.txt, Internet draft, June 1997.
16. M. Loukola, J. Ruutu, and K. Kilkki, "Dynamic RT/NRT PHB Group," draft-loukola-dynamic-00.txt, Internet draft, November 1998.

17. C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," in *Proceedings of ACM SIGCOMM'99*, Cambridge, MA, September 1999, pp. 109–120.
18. T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "Relative Delay Differentiation and Delay Class Adaptation in Core-Stateless Networks," in *Proceedings of IEEE INFOCOM'00*, Tel-Aviv, Israel, March 2000.
19. Y. Moret and S. Fdida, "A Proportional Queue Control Mechanism to Provide Differentiated Services," in *International Symposium on Computer System*, Belek, Turkey, October 1998.
20. "WTP packet scheduler for ns2," http://www.cis.udel.edu/∼dovrolis/ns-WTP.tar.
21. A. Feldmann, A. C. Gilbert, and W. Willinger, "Data networks as cascades: Investigating the multifractal nature of the Internet WAN traffic," in *Proceedings of ACM SIGCOMM'98*, Vancouver, Canada, August 1998, pp. 25–38.
22. "Diffserv Model for the ns2 simulator," http://www7.nortel.com:8080/CTL/.

Service Differentiation in ECN Networks Using Weighted Window-Based Congestion Control for Various Packet Marking Algorithms[*]

Vasilios A. Siris[1], Costas Courcoubetis[1,2], and George Margetis[1]

[1] Institute of Computer Science (ICS)
Foundation for Research and Technology - Hellas (FORTH)
P.O. Box 1385, GR 711 10 Heraklion, Crete, Greece
{vsiris, courcou, gmarget}@ics.forth.gr

[2] Dept. of Informatics, Athens University of Economics and Business, Greece

Abstract. We investigate the service differentiation, in terms of average throughput, and the performance achieved using weighted window-based congestion control in networks supporting Explicit Congestion Notification (ECN). Our results show how service differentiation, queueing delay, and average throughput are affected by the increase and decrease rules of the end-system congestion control algorithms, and how they depend on the marking algorithms operating in the routers. The end-system algorithms we investigate include algorithms that achieve an average throughput proportional to some willingness-to-pay, and the marking algorithms include RED, virtual queue marking, and load-based marking.

1 Introduction

The Transmission Control Protocol (TCP) [8] has played an important role in the Internet's growth. With TCP, however, all connections with the same round trip time receive the same average throughput in the equilibrium; hence, it can not support service differentiation. In the highly competitive telecommunications market, the ability to provide, in a flexible and efficient way, differentiated services will be extremely important. Moreover, new active queue management algorithms [3] combined with Explicit Congestion Notification (ECN) [16] open up new possibilities for alternative approaches to congestion control.

One approach for supporting service differentiation, which is followed by the differentiated services (DiffServ) architecture, is to add mechanisms inside the network. Drawbacks to this approach are the increased complexity, compared to the Internet today, and the need for co-operation among the routers in order to provide end-to-end services. An alternative approach has emerged [6,10,12], which suggests that service differentiation, as well as efficient and stable network operation and growth to meet increasing demand, can be achieved by a network with a simple feedback mechanism, such as ECN marking, that informs users

[*] This work is supported in part by the European Commission under IST project M3I "Market Managed Multiservice Internet" (IST-1999-11429).

of the congestion cost their traffic is incurring. The users then react to these congestion signals using some rate control algorithm; moreover, the network provider can give users the incentive to react to the congestion signals, hence use the network, according to their actual needs by charging a small fixed price per mark.

In this paper we consider the latter approach and assume a network where congestion feedback is in the form of ECN marks, and no losses occur. For such a network we investigate a family of weighted window-based congestion control algorithms, which we will refer to as willingness-to-pay (WTP) algorithms, that are a generalization of the algorithm first proposed in [6]. With WTP, the congestion window follows a *single multiplicative decrease*: the rate of ECN marks is roughly[1] proportional to the sending rate and the congestion window is decreased by some *fixed* amount for each ECN mark received. On the other hand, TCP follows a *double multiplicative decrease*: both the rate of ECN marks and/or losses is proportional to the sending rate and the congestion window is *halved* upon detection of congestion. The willingness-to-pay algorithms achieve an average rate that, depending on the marking algorithm in the routers, is proportional to some weight, or willingness-to-pay, and inversely proportional to the marking probability inside the network. As we discuss later, the willingness-to-pay parameter can affect the amount by which the congestion window increases when an acknowledgement without an ECN mark is received, or the amount it decreases when an acknowledgement with an ECN mark is received, or both.

Our objective is to investigate, using simulation, the service differentiation achieved by the WTP congestion control algorithms, and how the performance of these algorithms is affected by the marking algorithms operating in the routers. With service differentiation we refer to the ability of the end-system algorithms, working in conjunction with the marking algorithm in routers, to offer different throughput to connections with different weights or willingness-to-pay values. The three marking algorithms we investigate are the RED (Random Early Detection) algorithm [5], modified such that packets are only marked and never dropped, the virtual queue algorithm [6], that marks packets depending on the overflow of a virtual queue whose buffer and capacity are some percentage of the actual buffer and capacity of the link, and a load-based algorithm where the marking probability is a linear function of the link utilization measured over some time interval.

Our major results are summarized as follows:

- With the virtual queue algorithm, where the marking probability depends on the traffic burstiness, service (i.e., average throughput) differentiation depends on the characteristics of the congestion control algorithm, and in particular on the increase and decrease rules.
- With appropriate tuning, all three marking algorithms can exhibit the same marking probability for a range of average load values. As discussed in [10],

[1] As we discuss in more detail later, this depends on the marking algorithm operating in the router.

the marking probability as a function of the average load gives information regarding the convergence and stability behaviour of the marking algorithm.
- Both RED and load-based marking, where packet marking is probabilistic, have smaller average queueing delay and delay variation compared to virtual queue marking.
- Comparison of WTP congestion control with MulTCP [2], which follows TCP's double multiplicative decrease but supports service differentiation, shows that MulTCP can be less fair and can achieve lower link utilization compared to WTP.

The rest of this paper is structured as follows: In Section 2 we summarize the theoretical background of our work. In Section 3 we present the willingness-to-pay class of congestion control algorithms, and in Section 4 we discuss the three marking algorithms considered in our investigations. In Section 5 we present and discuss the results from our simulation experiments. Finally, in Section 6 we discuss related work and in Section 7 we present some concluding remarks.

2 Theoretical Background

In this section we summarize the theoretical background of our work as developed by Kelly and other researchers [9,11,10].

Consider a network of J resources [9,11]. Each resource (link) j marks packets with some probability $p_j(y_j)$, where y_j is the aggregate arrival rate at resource j.

Let R be the set of routes, or connections, active in the network. Each connection r updates its rate x_r according to the equation

$$\frac{dx_t(t)}{dt} = \kappa_r \left(w_r - x_r(t) \sum_{j \in r} \mu_j(t) \right) , \tag{1}$$

where w_r is a weight, κ_r is a constant that controls the rate of convergence, and

$$\mu_j(t) = p_j \left(\sum_{s : j \in s} x_s(t) \right) ,$$

is the marking probability at resource j. With (1), a connection r adjusts its sending rate x_r so that the rate of marks it receives $x_r(t) \sum_{j \in r} \mu_j(t)$ becomes equal to the weight w_r. Hence, if the network charges a fixed amount for each mark returned to the end-system, then w_r represents the willingness-to-pay for connection r.

Assume that the marking probability $p(y_j(t))$ depends on the cost $C_j(y_j)$ incurred on resource j as follows

$$p_j(y_j) = \frac{dC_j(y_j)}{dt} .$$

It can be shown that, if $C_j(y_j(t))$ is differentiable and feedback is instantaneous, then the above system converges to a point that maximizes

$$\sum_{r \in R} w_r \log x_r - \sum_{j \in J} C_j \left(\sum_{s: j \in s} x_s(t) \right) .$$

If $U_r(x_r) = w_r \log x_r$ is taken to be the utility for a connection r, then the last expression represents the social welfare of the system (resources and connections).

The above results on social welfare maximization can be generalized for the case where a connection r has a utility with a general form $U_r(x_r)$, if the willingness-to-pay changes smoothly according to

$$w_r(t) = x_r(t) U_r'(x_r(t)) .$$

The rate-based control algorithm given by (1) can be approximated using a window-based, self-clocking algorithm, similar to TCP's congestion avoidance algorithm, if a connection's congestion window *cwnd* is updated with the reception of an acknowledgement using, see [6],

$$cwnd{+}= \bar{\kappa} \left(\frac{\bar{w}}{cwnd} - f \right) , \tag{2}$$

where $\bar{\kappa} = \kappa T$ is the gain factor per round trip time, $\bar{w} = wT$ is the willingness to pay per round trip time, and f equals 1 if the acknowledgement contains a mark or 0 if it does not. Note that (1) and its window-based version (2) follows a single multiplicative decrease, since the rate of marks is (roughly) proportional to the sending rate and the congestion window is decreased by some fixed amount for each ECN mark received. This is unlike TCP's congestion avoidance algorithm which follows a double multiplicative decrease, since both the rate of marks and the decrease of the congestion window for each mark (in particular, it is halved) is proportional to the sending rate.

The above results assumed instantaneous feedback. Next we summarize the approach of Kelly [10] for taking feedback delay into account, for the case of a single resource when all connections have the same gain parameter κ and under the assumption that the queueing delay represents a small fraction of the round trip time T (feedback delay), which is fixed and common for all connections. Let $x(t) = \sum_r x_r(t)$ and $w = \sum_r w_r$. Summing (1) for all connections and taking the time lag into account we get

$$\frac{dx(t)}{dt} = \kappa \left(w - x(t-T) p(x(t-T)) \right) . \tag{3}$$

The linear delay equation

$$\frac{du(t)}{dt} = -\alpha u(t-T) \tag{4}$$

converges to zero as t increases if $\alpha T < \pi/2$. Moreover, the convergence is non-oscillatory if $\alpha T < 1/e$. Letting $x(t) = x + u(t)$, where x is the equilibrium of (3), and linearizing equation (3) about x, we get equation (4) with $\alpha = \kappa(p + xp')$, where p, p' are the marking probability and its derivative at rate x. Hence, the equilibrium of (3) is stable and the convergence is non-oscillatory if

$$\kappa T(p + xp') < e^{-1} .$$

3 Weighted Window-Based Congestion Control

In this section we describe a class of window-based congestion control algorithms that are a generalization of the algorithm given by (2) and first presented in [6].

The congestion window *cwnd* is updated using

$$cwnd{+}= \bar{\kappa}\left(\frac{\bar{w}_{\rm inc}}{cwnd} - \frac{f}{\bar{w}_{\rm dec}}\right) ,$$

hence, the average change of the rate per unit time is

$$\frac{\bar{\kappa}\left(\frac{\bar{w}_{\rm inc}}{cwnd} - \frac{p}{\bar{w}_{\rm dec}}\right)/T}{T/cwnd} = \frac{\bar{\kappa}}{T}\left(\frac{\bar{w}_{\rm inc}}{T} - \frac{xp}{\bar{w}_{\rm dec}}\right) , \tag{5}$$

where, as before, $\bar{\kappa}$ is the gain factor per round trip time and $f = 1$ or 0 if the acknowledgement contains or does not contain an ECN mark, respectively. From the last equation we can deduce that for connections with the same round trip time, the average rate is proportional to $\bar{w}_{\rm inc}\bar{w}_{\rm dec} = \bar{w}$. For $\bar{w}_{\rm inc} = \bar{w}, \bar{w}_{\rm dec} = 1$ we have the proportional increase algorithm given by (2). For $\bar{w}_{\rm inc} = 1, \bar{w}_{\rm dec} = \bar{w}$ we have the inversely proportional decrease algorithm given by

$$cwnd{+}= \bar{\kappa}\left(\frac{1}{cwnd} - \frac{f}{\bar{w}}\right) . \tag{6}$$

The values of $\bar{w}_{\rm inc}$, $\bar{w}_{\rm dec}$ represent a trade-off between the aggressiveness in probing for available bandwidth, hence in reaching the steady state, and the stability and size of the fluctuations around the average congestion window (equivalently, around the average throughput).

Substituting $\kappa = \bar{\kappa}/T$ and $w_{\rm inc} = \bar{w}_{\rm inc}/T$ in (5), summing for all connections, and taking the time lag into account we get

$$\frac{dx(t)}{dt} = \kappa\left(w_{\rm inc} - \frac{x(t-T)p(x(t-T))}{\bar{w}_{\rm dec}}\right) ,$$

which, using the results of Kelly [10] that are summarized in the previous section, is stable and the convergence is non-oscillatory if

$$\frac{\kappa T(p + xp')}{\bar{w}_{\rm dec}} < e^{-1} . \tag{7}$$

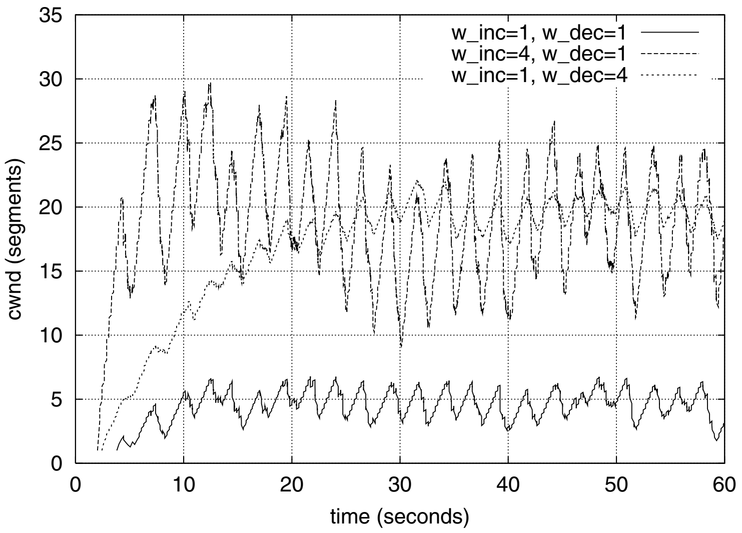


Fig. 1. For $\bar{w}_{\rm inc} = 4, \bar{w}_{\rm dec} = 1$ the congestion window reaches the equilibrium faster, but fluctuations are larger. On the other hand, for $\bar{w}_{\rm inc} = 1, \bar{w}_{\rm dec} = 4$ the congestion window reaches the equilibrium slower, but fluctuations are smaller. Both connections achieve approximately the same average congestion window, hence throughput.

The last equation is easier to satisfy for larger values of $\bar{w}_{\rm dec}$.

The trade-off between rate of convergence and magnitude of fluctuations in the equilibrium is depicted in Figure 1. As noted above, increasing $\bar{w}_{\rm dec}$, while keeping the product $\bar{w}_{\rm inc}\bar{w}_{\rm dec} = \bar{w}$ constant, results in smaller fluctuation around the equilibrium. Smaller fluctuations can be advantageous for streaming applications, which typically perform better when their sending rate does not change abruptly.

As indicated by (7), convergence also depends on the marking probability p and its derivative p': In general, a smaller value of p and p' results in faster convergence, but, on the other hand, will also result in larger fluctuations around the equilibrium [11]. As we investigate in Section 5, the marking probability depends both on the congestion control algorithm and the marking probability operating in the routers.

4 Packet Marking Algorithms

In this section we describe the three marking probability algorithms, namely Random Early Detection (RED), virtual queue marking, and load-based marking, that we investigate in Section 5.

4.1 Random Early Detection (RED)

With RED [5], the marking probability is a piecewise linear function of the average queue length $\bar{q}$, which is estimated using exponential averaging with weight factor w_q. Specifically, if $\bar{q}$ is smaller than some minimum queue length min_{th}, then the marking probability is zero. If $\bar{q}$ is between min_{th} and some maximum queue length max_{th}, then the marking probability ranges linearly from 0 to some maximum value max_p. Finally, if $\bar{q}$ is above max_{th} then packets are dropped. The "gentle_" modification[2] of the original RED algorithm suggests to vary the marking probability from max_p to 1, when $\bar{q}$ is between max_{th} and $2max_{th}$.

In the investigations of the next section we use the RED algorithm with the "gentle_" variation and modified so that packets are never dropped but only marked.

4.2 Virtual Queue Marking

The virtual queue marking algorithm [6] presents an early warning of congestion. The algorithm maintains a virtual buffer of size θB serviced at rate θC, where B, C are the actual buffer and capacity of the output link, respectively. Note that B is not necessarily the total buffer of the output link, but can be some value that corresponds to a maximum target delay. The algorithm marks all packets that arrive at the link from the time a loss occurs in the virtual buffer until the first time the virtual buffer becomes empty; this period is called the busy period of the virtual buffer. Note that there are other variations for when to mark packets: For example, the algorithm in [7] marks incoming packets when they cause the virtual queue to overflow.

As we will see in the next section, a property of the virtual queue algorithm is that it differentiates flows based on their burstiness. Another property, demonstrated in experiments that are not presented in this paper, is that it does not avoid cases of synchronization of phases in closed-loop congestion control algorithms, such as those observed for drop-tail routers [4], which can result in some connections achieving very large average throughput, and other connections achieving very small throughput. Indeed, one motivation for introducing RED in routers was to avoid such effects [4,5].

4.3 Load-Based Marking

The third marking algorithm we investigate is load-based marking. According to this algorithm, the marking probability is a piecewise linear function of the average load (utilization), which is measured over some time interval. In contrast, with RED the marking probability is a piecewise linear function of the average queue length. With the load-based algorithm, the marking probability is zero

[2] See 'Recommendation on using the "gentle_" variant of RED', S. Floyd, 2000, `http://www.aciri.org/floyd/red/gentle.html`

when the average load is less than some minimum value ρ_0. For values of the load ρ larger than ρ_0 the marking probability is given by $\min\{\alpha(\rho - \rho_0), 1\}$. Hence, the algorithm has three parameters: the minimum load ρ_0, the parameter α that controls the slope of the marking probability, and the averaging interval t_{avg}.

The averaging interval determines how quickly the algorithm adjusts the marking probability to changes of the load. Moreover, the interval determines the timescales over which congestion is detected, hence the timescales over which traffic burstiness affects the marking probability.

Load-based marking can be particularly appropriate in cases where there is no buffering, such as Code Division Multiple Access (CDMA) wireless networks and Ethernet local area networks, since unlike the previous two marking algorithms, it does not rely on the queue length as a measure of congestion.

5 Simulation Results and Discussion

In this section we present and discuss simulation results investigating the interaction of the willingness-to-pay (WTP) congestion control algorithms discussed in Section 3, and in particular the proportional increase (2) and the inversely proportional decrease (6) algorithms, and the three marking algorithms discussed in Section 4.

The specific issues we investigate include the following:

- Service differentiation: Dependence of the average throughput achieved by different connections on their weight or willingness-to-pay, and how this is affected by the marking algorithm.
- Dependence of the marking probability on the average load: As discussed in Section 2, this dependence gives information regarding the stability and convergence behaviour of the whole system, i.e., the congestion control algorithms in the end-systems and the marking algorithms in the routers.
- Queueing delay and link utilization: In particular, we investigate the queueing delay for different marking algorithms, when the average link utilization is the same.
- Comparison of the WTP and MulTCP algorithms: Recall that WTP follows a single multiplicative decrease, whereas MulTCP follows a double multiplicative decrease, similar to normal TCP.

The simulations were performed using the ns-2 simulator [17]. The topology of the simulated network is shown in Figure 2. Since our focus is on investigating the interaction of the congestion control algorithms at the end-systems and the marking algorithms at the routers, we consider link parameters that do not result in packet loss, since this would generate retransmissions, which would affect the throughput measurements. Also, we focus on the congestion avoidance phase, hence our measurements begin after some time interval has elapsed from the start of each experiment.

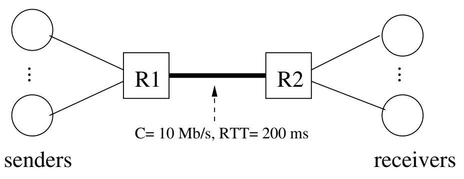


Fig. 2. Network topology used in the simulation experiments. The link connecting the two routers is the bottleneck.

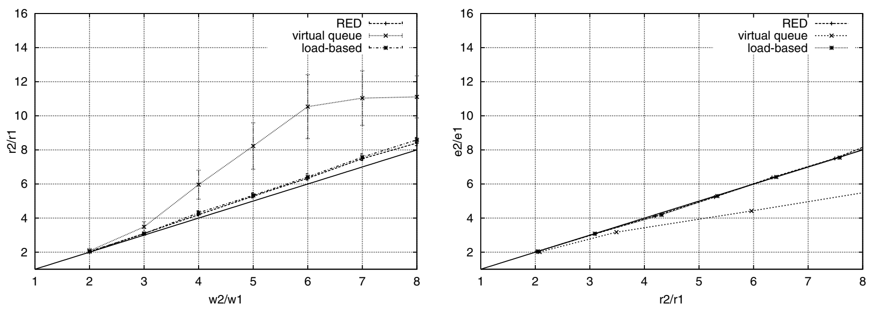
5.1 Service Differentiation

Figure 3(a) shows the average ratio of throughput for different ratios of willingness-to-pay values. Also shown is the 95% confidence interval. The results were obtained from 10 independent runs of the experiment with the same parameters. Each connection carried data from a long ftp transfer, and the start time of each connection was selected randomly from the interval [0, 5] seconds. Finally, the throughput was computed for the interval [60, 180] seconds.

Observe that the service differentiation achieved by both RED and load-based marking is roughly the same. Moreover, the ratio of the average throughput is very close to the ratio of willingness-to-pay. Note, however, that the two curves are slightly above the diagonal; this is attributed to the window-based nature of the congestion control algorithm, since in results not shown here, the ratio of the congestion windows is approximately equal to the ratio of willingness-to-pay.

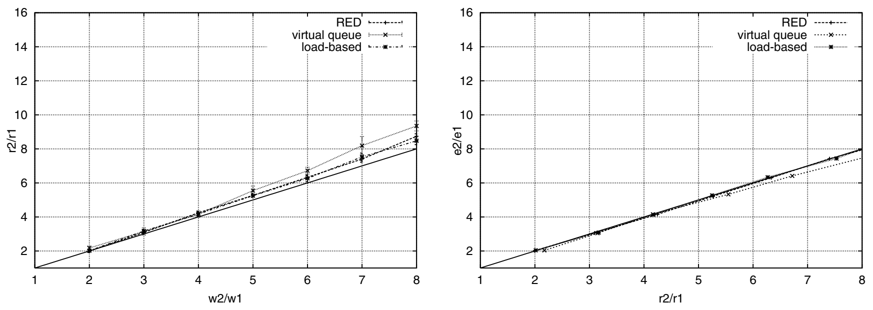
On the other hand, in the case of virtual queue marking observe that for large values of the ratio of willingness-to-pay, the ratio of throughput is larger than the ratio of willingness-to-pay. Moreover, as indicated by the confidence interval, the fluctuations in service differentiation are also much larger compared to the fluctuations for the other two marking algorithms. Next we explain both of these observations.

The first observation can be explained with the help of Figure 3(b), which shows the ratio of marks as a function of the ratio of average throughput. The figure shows that a smaller percentage of the packets belonging to a connection with a larger willingness-to-pay are marked, i.e., the marking probability is smaller for a connection with a larger willingness-to-pay. This is due to the combination of the following two factors: first, the connection with a smaller willingness-to-pay sends a smaller number of segments in one round trip time and second the segments are typically sent back-to-back; the latter being a property of any window-based control mechanism. As a result, the connection with a smaller willingness-to-pay produces a burstier traffic stream compared to the connection with a larger willingness-to-pay. Burstier traffic streams, however, are more difficult for a multiplexer (link) to handle, hence require more bandwidth than their average rate. The virtual queue marking algorithm has the property of differentiating streams based on their burstiness. Hence, the algorithm marks a higher percentage of packets belonging to the burstier stream, which, as explained above, is the stream with smaller willingness-to-pay.



(a) ratio of throughput (b) ratio of received marks

Fig. 3. Service differentiation using proportional increase WTP. $\bar{\kappa} = 0.5$. $C = 10$ Mbps, $RTT = 200$ msec, $N = 10$. RED: $min_{th} = 5, max_{th} = 15, max_p = 0.1, w_q = 0.002$. VQ: $vb = 0.95 \cdot 30, vc = 0.95 \cdot 10$ Mbps. LB: $t_{avg} = 0.5$ s, $\rho_0 = 0.6, \alpha = 0.71$.

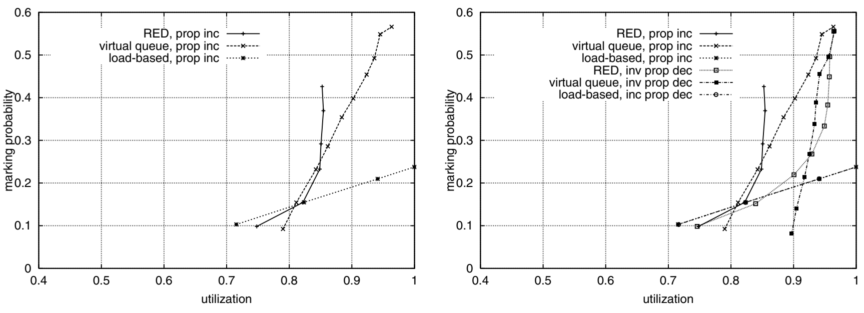


(a) ratio of throughput (b) ratio of received marks

Fig. 4. Service differentiation using inversely proportional decrease WTP. $\bar{\kappa} = 0.5$. $C = 10$ Mbps, $RTT = 200$ msec, $N = 10$. RED: $min_{th} = 5, max_{th} = 15, max_p = 0.1, w_q = 0.002$. VQ: $vb = 0.95 \cdot 30, vc = 0.95 \cdot 10$ Mbps. LB: $t_{avg} = 0.5$ s, $\rho_0 = 0.6, \alpha = 0.71$.

The above is not the case when the end-systems implement inversely proportional decrease WTP, as shown in Figure 4(a). In this case, and for the timescales in which packet marking occurs, the traffic from WTP connections with a different willingness-to-pay exhibit roughly the same burstiness, hence are not differentiated and the service differentiation for virtual queue marking is close to that of RED and load-based marking. This is also illustrated in Figure 4(b), which shows that in this case the ratio of marks for all three algorithms is roughly proportional to the ratio of willingness-to-pay values.

Next we explain the second observation made from Figure 3(a), namely the larger variations of service differentiation achieved with virtual queue marking compared with to that of RED and load-based marking. With the latter two algorithms packet marking is probabilistic, hence marks tend to be spread out; this results in smoother changes of the congestion window. On the other hand, the virtual queue algorithm tends to produce bursts of marks, which result in larger fluctuations of the congestion window.



(a) Proportional increase (b) Inversely proportional decrease

Fig. 5. Marking probability as a function of load for proportional increase WTP: $\bar{\kappa} = 0.5, \bar{w}_{\text{inc}} = 1$ and $6, \bar{w}_{\text{dec}} = 1$, and inversely proportional decrease WTP: $\bar{\kappa} = 0.5, \bar{w}_{\text{inc}} = 1, \bar{w}_{\text{dec}} = 1$ and 6. $C = 10$ Mbps, $RTT = 200$ msec. RED: $min_{th} = 2, max_{th} = 6, max_p = 0.1, w_q = 0.02$. VQ: $vb = 0.95 \cdot 30, vc = 0.95 \cdot 30$ Mbps (left figure). LB: $t_{avg} = 0.5$ s, $\rho_0 = 0.6, \alpha = 0.71$.

5.2 Marking Probability

Next we investigate the marking probability as a function of the average load, for the three marking algorithms we consider. Note that this probability depends, in addition to the load, also on the burstiness of traffic, which in turn is affected by both the congestion control algorithm in the end-systems and the marking algorithm in the routers. The shape of this curve, as discussed in Section 2 and [10], gives an indication of the convergence and stability behaviour of the particular marking algorithm.

Figures 5(a) and (b) show the marking probability as a function of average load for the three marking algorithms and for the proportional increase and inversely proportional decrease algorithms running on the end-systems. Observe that for both virtual queue and load-based marking, there is a linear dependence of the marking probability on the link utilization. On the other hand, the marking probability for RED is convex and becomes steep for large values of the load, although as we will see later this is not always the case. Figure 5(b) shows that for both virtual queue marking and RED, the end-system congestion control algorithm has an affect on the marking probability. Indeed, for the inversely proportional decrease congestion control algorithm, the same marking probability is achieved for a higher utilization. This is expected, since the inversely proportional decrease algorithm results in smoother traffic, as discussed in Section 3, hence can achieve a higher utilization, for the same marking probability.

Figures 5(a) and (b) were obtained for some arbitrary[3] selection of parameters for each marking algorithm. Figure 6 shows that these parameters can be chosen such that the marking probability curves for the three algorithms become

[3] For RED we used parameters suggested in the literature. Our results indicate that the rules for tuning of RED in the case of WTP congestion control are not the same as in the case of TCP.

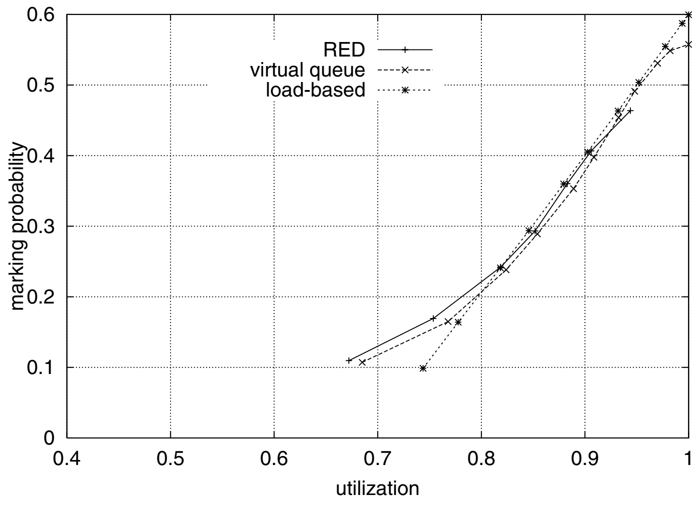


Fig. 6. Marking probability as a function of link utilization. $\bar{\kappa} = 0.5, \bar{w}_{\text{inc}} = 1$ and $6, \bar{w}_{\text{dec}} = 1$. $C = 10$ Mbps, $RTT = 200$ msec. RED: $min_{th} = 3, max_{th} = 9, max_p = 0.95, w_q = 0.002$. VQ: $vb = 0.95 \cdot 20, vc = 0.95 \cdot 10$ Mbps. LB: $t_{avg} = 0.5$ s, $\rho_0 = 0.7, \alpha = 2$.

very close. This indicates that, at least in terms of the macroscopic convergence and stability properties, all marking algorithms have similar behaviour when their parameters are tuned appropriately. Note, however, that these curves are averages, and as we will see in the next subsection, do not give a complete picture of the queueing delay for each marking algorithm.

5.3 Queueing Delay

In the previous subsection we saw that the parameters of the marking algorithms we consider can be tuned so that the average marking probability is roughly the same for a range of average utilization values. Since these are average values, they do not present a complete picture of the queueing behaviour. Table 1 shows the queueing delay for the three marking algorithms when the average utilization is kept the same. The bottom three rows in the table show that the queueing delay is smaller and exhibits smaller fluctuations for RED and load-based marking, compared to virtual queue marking. Again, this is expected since, as discussed in Section 5.1, due to their probabilistic marking, both RED and load-based marking result in smoother traffic. The first two rows show that for the same marking algorithm the queueing delay depends on the parameters of the marking algorithm, even when the utilization is kept the same

5.4 Effect of Marking Algorithm Parameters

In addition to their performance, how easy or difficult it is to tune the parameters of a marking algorithm is equally, if not more, important. Towards this end, in

Table 1. Queueing delay (in msec). Long ftp flows, $\rho \approx 0.90$. The bottom three rows show that the queueing delay is smaller for RED and load-based marking, compared to virtual queue marking.

marking algorithm	average	maximum	std. dev.
RED ($min_{th} = 5, max_{th} = 15,$ $max_p = 0.10, w_q = 0.002$)	26.9	89.6	26.4
RED ($min_{th} = 3, max_{th} = 9,$ $max_p = 0.95, w_q = 0.002$)	5.3	22.4	3.6
virtual queue ($vb = 0.95 \cdot 20, vc = 0.95 \cdot 10$ Mbps)	8.6	39.4	6.2
load-based ($t_{avg} = 0.5$ s, $\rho_0 = 0.7, \alpha = 2$)	6.3	38.4	4.8

this subsection we provide some insight on how the parameters affect the marking probability function.

Figure 7(a) shows how the marking probability for the virtual queue marking algorithm is affected by the factor θ, which determines the buffer and capacity of the virtual queue. Observe that as θ increases, the marking probability shifts to the right, i.e., for larger θ, the same utilization corresponds to a lower marking probability. Also, observe that the factor has a very small effect on the slope of the curve. Finally, Figure 7(b) shows that a larger buffer results in a steeper curve for the marking probability. Indeed, for small utilizations, the marking probability is higher for a smaller buffer, since the buffer overflows more frequently. On the other hand, for large utilizations the effect is the opposite, i.e., the marking probability is higher for a larger buffer, because now the dominating effect is the larger overflow (marking) periods, since a larger buffer needs more time to empty.

For load-based marking the effect of its parameters is straightforward: The parameter α affects the slope of the marking probability curve, ρ_0 (for constant α) affects the marking probability for a given average throughput, and the averaging interval affects how fast the algorithm responds to changes of network load and the timescales over which traffic burstiness affects the marking probability. Moreover, the averaging interval affects the maximum queue backlog that appears when the congestion level changes.

Finally, for RED we have observed that when max_p is small, then the marking probability function is convex, as in Figure 5. On the other hand, when max_p approaches 1, then the marking probability function tends to be linear, as in Figure 6. In general, we have found that the rules for tuning the parameters of RED in the case of WTP congestion control are not the same as the rules in the case of TCP.

5.5 Comparison of WTP and MulTCP

Figure 8 compares willingness-to-pay congestion control with MulTCP [2]. The latter follows TCP's double multiplicative decrease, but provides support for service differentiation. Figure 8(a) shows that the ratio of average throughput is

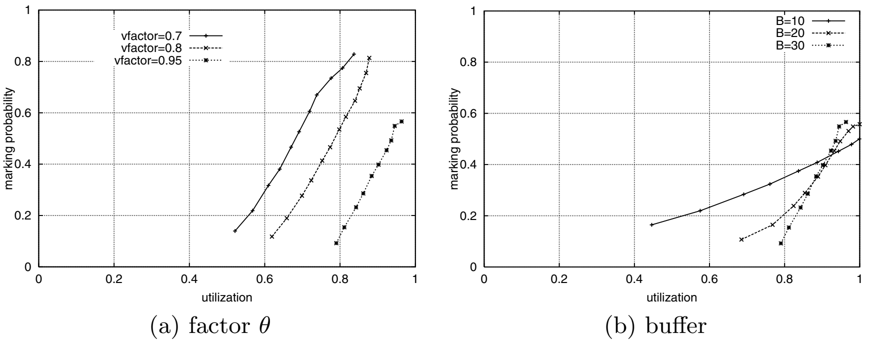


Fig. 7. Effect of the virtual queue factor θ and the buffer size. $C = 10$ Mbps, $RTT = 200$ msec, $\bar{w}_{\text{inc}} = 1$ and $6, \bar{w}_{\text{dec}} = 1, \bar{\kappa} = 0.5$.

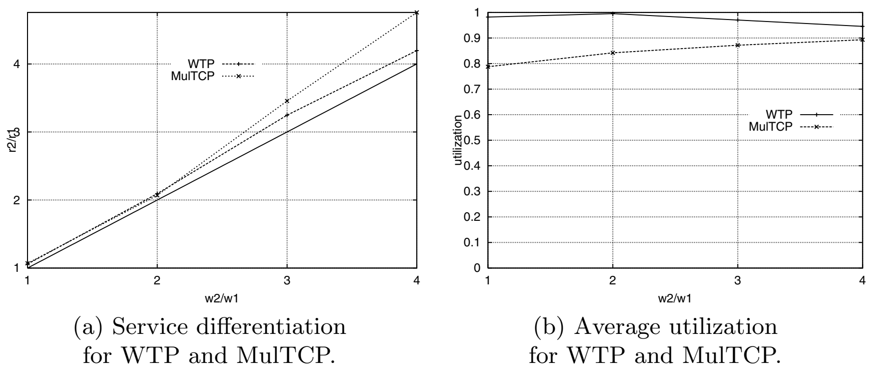


Fig. 8. Comparison of WTP and MulTCP. WTP: $\bar{\kappa} = 0.5$. $C = 10$ Mbps, $RTT = 200$ msec, $N = 10$. RED Marking: $min_{th} = 5, max_{th} = 15, max_p = 0.1, w_q = 0.002$.

higher than the ratio of weights for MulTCP, indicating that MulTCP favours connections with larger weights.

Figure 8(b) compares the utilization achieved with WTP and MulTCP, when the same number of connections are multiplexed in a link with RED marking. Observe that for small weights, WTP achieves a higher utilization compared to MulTCP. As the weights increase, however, the difference between the two congestion control algorithms is smaller. We conjecture that this is due to the following: For MulTCP, connections are bursty even for small weights, due to the proportional decrease of the congestion window when an ECN mark is received; moreover, as the weights increase, the resulting increase of aggressiveness helps connections achieve a higher average throughput. On the other hand, for small weights WTP connections produce smooth traffic, hence the average utilization is high; when the weights increase, however, connections become burstier, and as a result the average utilization decreases.

6 Related Work

Next we discuss some related work. Note that this is not an exhaustive survey of all the work in the area.

The authors of [15] investigate the limitations of using double proportional decrease algorithms for achieving service differentiation. Their results are in agreement with the results of Section 5.5. They also investigate an algorithm where the congestion window is adjusted based on the percentage of lost packets. In contrast, our investigations of the WTP algorithms are for networks with zero loss. The loss adaptive algorithm in [15] achieves in the steady state an average throughput inversely proportional to the loss probability. This is similar to the dependence of the average throughput on the marking probability with the WTP algorithms investigated in this paper. Comparison of the two algorithms in terms of throughput and delay is an area for further investigation.

The authors of [14] investigate the convergence and steady state behaviour, using both simulation and dynamical system modelling, of the rate-based congestion control algorithm given by (1) and when routers implement threshold-based marking, i.e., all packets are marked when the queue length exceeds a threshold. Additionally, they discuss issues regarding the timescales for a connection's rate to reach equilibrium in relation to the timescales of the inter-arrival time between new connections and the duration of each connection.

The author of [18] discusses the fairness of marking algorithms, using ideas from large deviations theory and economics. A number of marking algorithms that have been proposed in the literature are investigated, and a threshold-based marking algorithm that marks all packets in the queue when the queue length exceeds a threshold is proposed. According to the algorithm, the threshold is adjusted adaptively based on the number of packets that have contributed to the queue exceeding the threshold, hence packet marks represent the incremental cost for sending an additional packet, where cost is taken to be the number of packets exceeding the threshold level.

The authors of [13] investigate, using a fluid model, the decentralized selection of the marking rate at each router of a network in order to achieve loss-free operation. The convergence of the scheme is investigated using a timescale decomposition of the resulting system into a slow and fast system model.

7 Concluding Remarks

We have investigated, using simulation, the service differentiation and performance achieved with end-to-end weighted window-based congestion control algorithms, and how this is affected by the marking algorithms implemented in the routers. The congestion control algorithms investigated include the willingness-to-pay (WTP) algorithms that follow a single multiplicative decrease of the congestion window upon congestion. Our results show that, for networks with no losses supporting Explicit Congestion Notification (ECN), the WTP algorithms can be fairer and lead to higher utilization compared to MulTCP, which supports service differentiation and follows a double multiplicative decrease, similar to normal TCP. Note that in cases where the loss rate tends to be a large, algorithms that decrease the congestion window by a large amount, e.g., by half as in the case of TCP and MulTCP, might be more appropriate, since a large decrease of the congestion window would result in a lower loss rate, hence a smaller number of retransmissions.

The marking algorithms investigated include RED, virtual queue marking, and load-based marking. Our results show that service differentiation and queueing delay can be worse for the virtual queue algorithm, where packet marking occurs at the timescales of overflow of a virtual buffer, compared to RED and load-based marking where the marking probability depends on averages (queue length for RED and load for load-based marking). On the other hand, the virtual queue marking algorithm can differentiate flows according to their burstiness.

Our investigations were limited to the case where all connections had the same round trip time. When connections have different round trip times, which would be the case in more complex network topologies than the one considered in this paper, the appropriate selection of parameters for marking algorithms, such as RED and load-based, which determine the marking probability based on some average quantities, presents difficulties. These difficulties concern the parameters, such as the exponential weight factor of RED and the averaging interval of load-based marking, which determine the time interval over which averaging is performed, hence how fast the end-systems are informed of changes of the congestion level inside the network. Setting these parameters to achieve optimal feedback for long round trip time connections might be too slow for short round trip time connections. On the other hand, setting the parameters to achieve optimal feedback for short round trip time connections might be too fast for long round trip time connections, thus leading to larger fluctuations of the queue and instability. Moreover, the round trip time also affects the convergence and the average throughput in the equilibrium. The latter effect can be addressed, e.g., by having the increase or decrease of the congestion window upon the reception of an acknowledgement be proportional or inversely proportional, respectively, to the round trip time.

The willingness-to-pay and packet marking algorithms investigated in this paper have been implemented in a real test-bed comprising of workstations running FreeBSD[4]. Preliminary results from experiments conducted over the test-bed are reported in [1].

An issue for further investigation is how to automate the tuning of the parameters (adaptation) of the marking algorithms. Motivation for such automated adaptation is that, in agreement with the results regarding the tuning of RED parameters for TCP connections, no set of parameters are optimal for all possible traffic mixes and characteristics. Finally, another issue for further investigation is, in the case each mark is charged by a fixed price, how this price per mark affects the link utilization, and subsequently how to determine this price, given the aggregate demand, the congestion control algorithms in the end-systems, and the marking algorithms in the routers.

[4] The implementation of the algorithms in ns-2 and in FreeBSD are available at `http://www.ics.forth.gr/netgroup/publications/wtp_vq.html`

References

1. P. Antoniadis, C. Courcoubetis, G. Margetis, V. A. Siris, and G. D. Stamoulis. Efficient adaptation to dynamic pricing communicated by ECN marks: Scenarios for experimental assessment. In *Proc. of SPIE International Symposium on Information Technologies 2000 (Program on Internet Performance and Control)*, Boston, MA, 2000. Available at `http://www.ics.forth.gr/netgroup/publications/`.
2. J. Crowcroft and P. Oechslin. Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP. *Computer Communications Review*, 28(3):53–67, July 1998.
3. B. Braden *et al.* Recommendations on queue management and congestion avoidance in the internet. RFC 2309, April 1998.
4. S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3:115–156, 1992.
5. S. Floyd and V. Jacobson. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1:397–413, 1993.
6. R. J. Gibbens and F. P. Kelly. Resource pricing and congestion control. *Automatica*, 39:1969–1985, 1999.
7. R J. Gibbens, P. B. Key, and S. R. E. Turner. Properties of the virtual queue marking algorithm. In *Proc of 17th UK Teletraffic Symposium*, 2001.
8. V. Jacobson and M. J. Karels. Congestion avoidance and control. In *Proc. of ACM SIGCOMM'88*, pages 314–329, 1988.
9. F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, January 1997.
10. F. P. Kelly. Models for a self-managed Internet. *Philosophical Transactions of the Royal Society*, A358:2335–2348, 2000. Available at: `http://www.statslab.cam.ac.uk/~frank/`.
11. F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49, 1998.
12. P. B. Key and D. R. McAuley. Differential QoS and pricing in network: Where flow control meets game theory. *IEE Proceedings Software*, 146(2):39–43, March 1999.
13. S. Kunniyur and R. Srikant. A time scale decomposition approach to adaptive ECN marking. In *Proc. of IEEE INFOCOM'01*, Anchorage, AK, April 2001.
14. K. Laevens, P. Key, and D. McAuley. An ECN-based end-to-end congestion control framework: experiments and evaluation. Technical Report MSR-TR-2000-104, Microsoft Research, October 2000. Available at `http://research.microsoft.com/pubs`.
15. T. Nandagopal, K.-W. Lee, J.-R. Li, and V. Bharghavan. Scalable service differentiation using purely end-to-end mechanisms: features and limitations. In *Proc. of IEEE IWQoS'00*, Pittsburgh, PA, June 2000.
16. K. K. Ramakrishnan and S. Floyd. A proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, January 1999.
17. UCB/LBNL/VINT. Network Simulator - ns (version 2). `http://www-mash.cs.berkeley.edu/ns.html`.
18. D. Wischik. How to mark fairly. In *Proc. of Workshop on Internet Service Quality Economics*, MIT, December 1999.

Two-Differentiated Marking Strategies for TCP Flows in a Differentiated Services Network

Sung-Hyuck Lee, Seung-Joon Seok, Seung-Jin Lee,
Sung-Kwan Youm, and Chul-Hee Kang

Department of Electronics Engineering, Korea University
5-Ka, Anam-dong Sungbuk-ku,136-701 Seoul, Korea
{starsu, ssj, linuz, skyoum}@widecomm.korea.ac.kr
chkang@korea.ac.kr

Abstract. The saw-tooth like behaviors of TCP impact Assured Forwarding Service flows in a differentiated services network. Therefore, we argue the use of TCP-friendly building blocks(or modules) and fairness modules in the Differentiated Services architecture regarding this issue, and propose *Two Markers System(TMS)* that is able to properly mark packets and fairly share the bandwidth to each flow for their targeted sending rates. TMS has two marking modules that are placed on the source and the edge of a differentiated services network. For sources of the network, *the virtual source making modules* play important roles of reducing TCP impacts in the assured services and suitable marking packets. Next, in the edge of the network, *the edge embedded marking module* conducts new fairness policy based on the marking rate of flows from sources, so called *"marking rate-based fairness"*. Finally, we present simulation results to illustrate the effectiveness of TMS scheme over several parameters. That is, Two Markers System reduces TCP impacts over assured service and fairly shares the bottleneck link bandwidth of a network.

1 Introduction

The commodity Internet is based on the best-effort service model. In this model the service provider allocates bandwidth among all of the instantaneous customers as best it can, that is, all user packets compete equally for network resources. The service provider also attempts to serve all of them without making any explicit commitment as to rate or any other service quality. The best-effort model has been successful till now because the majority of the traffic on the Internet is TCP-based. The TCP end-to-end congestion control mechanism forces traffic sources to back off whenever congestion is detected in the network[8]. However, such dependence on the end systems' interaction is increasingly becoming unrealistic. Given the current best-effort model with FIFO queuing inside the network, it is relatively easy for non-adaptive sources to gain greater shares of network bandwidth and thereby starve other, well-behaved, TCP sources. A greedy source, for example, may simply continue to send at the same rate when faced with congestion while other TCP sources back off.

The diffserv approach is based on a set of simple mechanisms that treat packets differently according to the marking of the DS field in the IP header. Before entering in a DS domain, the field is marked with a certain value(or codepoint) that determines the treatment that should be supplied to the packet inside the domain. However, because of the limited amount of bits available for use in the DS field, the IETF's Diffserv Working Group has defined a small set of building blocks, called per-hop behaviors(PHBs) which are used by routers to deliver a number of services. Among the initial PHBs being standardized are the Expedited Forwarding(EF) and the Assured Forwarding(AF) PHBs. The EF PHB specifies a forwarding behavior in which packets see a very small amount of loss and a very low queuing delay. In order to ensure every packet marked with EF receives this service, EF requires every router to allocate enough forwarding resources so that the rate of incoming EF packets is always less than or equal to the rate which the router can forward them. The AF PHB group, on the other hand, specifies a forwarding behavior in which packets see a very small amount of loss, and consists of four, independently forwarded classes which have two or three drop preference levels. The idea behind AF is to preferentially drop best-effort packets and packets which are outside of their contract when congestion occurs.

In this paper, we consider a form of a better-than-best-effort service called the "Assured Service". The Assured Service follows expected capacity profiles which are statistically provisioned. Packets are treated preferentially according to the dropping probability applied to the best-effort queue. The assurance of service comes from the expectation that the traffic is unlikely to be dropped as long as it stays within the negotiated capacity profile. The building blocks of this service include a traffic marker at the edge of the domain, and a differentiated dropping algorithm in the core of the network. A packet of a flow is marked IN(in profile) if the temporal sending rate of the arrival time of the packet is within the contract profile of the flow. Otherwise, the packets are marked OUT(out-of-profile). The temporal sending rate of a flow is measured using TSM(Time Sliding Window) or token bucket control module. A differentiated dropping algorithm such as RIO(Random Early Detection with IN/OUT) is provided in the core routers of the network. In particular, the OUT packets are preferentially dropped upon evidence of congestion at the bottleneck before the IN packets. After dropping all incoming OUT packets, IN packets are discarded. With this dropping policy, the RIO network gives preference to IN packets and provides different levels of service to users based on their service contracts.

In [12], authors presented that the use of a simple token bucket marker for the above assured service results in TCP realizing the minimum assured rate. The authors attributed the cause of such behavior to TCP's complex response primarily to packet losses. TCP reacts to congestion by halving the congestion window(cwnd) and increases the window additively when packets are delivered successfully. Exponential decrease(halving the congestion window) is required to avoid congestion collapse and TCP treats a packet drop as an indication congestion[8]. However, in the differv network these additive-increase and multiplicative-decrease make it hard to protect the reservation rate. When TCP reacts to an OUT packet drop by halving its congestion window and increases additively, it may not reach its reservation rate.

In this paper, we take the problem into consideration between the transport control protocol(TCP) and the differentiated drop policies of the network in realizing the reserved throughputs, and propose newly modified scheme called, Two Markers System for improving the realization of target rates in a differentiated services network. In addition, the issue of fairness among users is important in this diffserv context since we expect that the differentiation in service will be based on some kind of pricing mechanism, thus it is necessary that the resource allocation be commensurate with how the service is priced. In this paper, we also argue how fairly the proposed scheme allocates bandwidth.

The rest of this paper is organized as follows: Section II reviews the state of the art in this area. Section III proposes Two Markers System to reduce TCP influences over assured services and consider fairness for a bottleneck link of a differentiated services network, and then, explores the proposed algorithms. Section IV and V present results using the above algorithms in simulated environments for responsive traffic flows such as TCP, and perform analysis for simulated results. Section VI concludes our work

2 Related Work

Recently, there have been a number of simulation work on marking and dropping policies that focused on a RED-based differentiated services approach.

In [13], Clark and Fang presented RIO(RED with In/Out) scheme and a remarking module that utilized an average sliding window rate estimator and sophisticated marking controller. The authors showed that source target rates could be assured in a simple capacity allocated network that depended on statistical multiplexing, and also presented that the in-profile portion of TCP flows were protected from issues such as Round Trip Time(RTT) and non-responsive UDP traffic flows.

In [1], [2], Feng et al. proposed adaptive priority marking schemes that focused on TCP-friendly mechanism in RIO scheme. The authors presented two different Marking mechanisms: source-integrated that the control engine, so called, packet marking engine (PME) is integrated with the host, and source transparent marking that PME is potentially external to the user. They showed results that the proposed scheme decreased TCP impacts over the Assured Service. They, however, didn’t consider marking scheme for aggregated flows.

In [12], through the medium of simulation work, Ibanez and Nichols showed that TCP didn’t realize the minimum assured rate over the Assured Service. The overall performance was affected by the bursty losses being experience by the OUT packets. They concluded that it was ambiguous how the Assured Service could be characterized quantitatively for TCP application, and an Assured Service couldn’t provide consistent rate guarantees.

In [9], Ikjun and Narasimha proposed and evaluated several schemes to improve the realization of throughput guarantees in the Internet that employed a differentiated services architecture. They showed that excess bandwidth of a network was not distributed proportional to target rates, and the impact of differences in RTTs could be reduced by the aggregation of sources and fair sharing of bandwidth at the edge routers.

In [7], Azeem et al. proposed that the TCP-friendly marker was designed to protect small windows from drops and maintain optimal spacing between IN and OUT packets. The authors, via simulation work, presented results showing that the TCP-friendly marker reduced the degree of token over-provisioning required to provide guarantees.

3 Two Markers System

In this section, we focus on several strategies used to mark packets in order to consider TCP dynamics and adapt fairness for sharing a bottleneck link of a network, and propose a modified marking scheme, so called, Two Markers System (TMS); the first marker(TMS_I) is located at sources of a network to adapt TCP congestion control algorithm, and the second marker(TMS_II) at edge to fairly mark the aggregated flow.

Marking strategies can be classified into three categories based on the criteria used for the marking. Module of devices can mark packets: (i) based on the state of all individual flows of an aggregation, called *flow marking*, (ii) based only on aggregation state, without any knowledge about individual flows, called *aggregation marking*, and (iii) based on a certain knowledge of individual flows, called *flow aware aggregation marking*. We will take newly improved marking strategy into consideration to complement the issue of the first and the second marking strategy.

If flow marking is performed on aggregated traffic, the device responsible for marking packets must deal with individual flows states. In this aspect, aggregation marking is easier to manage and is more suitable for customers who generate a enormous number of individual flows. A web-server is a good example of the customers with this characteristic. The number and the dynamics of short-term flows generated by this kind of customer can prevent devices from performing flow marking. The large number of states associated with metering needed to perform *flow marking* makes this strategy non-scalable. Furthermore, giving each flow a fraction of the target rate of the aggregated traffic can lead to an inefficient utilization of the reserved bandwidth. In this case, “idle” flows waste their shares while preventing “active” flows from increasing theirs. On the other hand, aggregation marking can introduce unfairness within aggregated flows, because the marking module marks the aggregated flows without any information on individual flows. The unfairness can be caused by different link bandwidth, different round-trip-times, different target rates, or different levels of congestion experienced by individual TCP flows within the network. In the flow aware aggregation marking category, markers may be not aware of how many flows may be marked, not to speak of whether the parameters associated with a particular flow are kept. However, the marker can maintain a partial state of the flows being marked, the bounded number of states can be a major advantage in certain scenarios.

Two Markers System considers the above problems: for leading to an inefficient utilization of the reserved bandwidth on *flow marking* and for introducing unfairness within aggregated flows on *aggregation marking*.

3.1 Concepts

What's the Two Markers System(TMS)? This system has two marking modules which are located in the source and at the edge of differentiated services network, and each marking module plays different roles to achieve the reservation rate and the target rate of Assured Services. Figure 1 illustrates the "Two Markers System" framework.

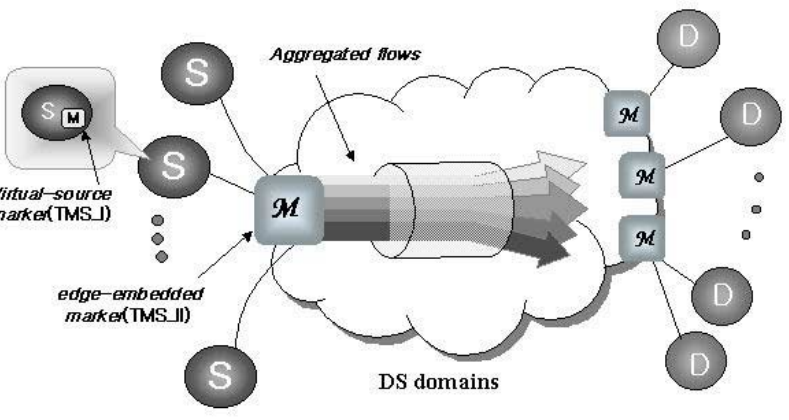


Fig. 1. The "Two Markers System" framework

First, a virtual source marking module(TMS_I) carries out two main roles. One is to control a flow and congestion, and the other is to give the marking probabilities to the edge marking module(TMS_II). Therefore, TMS_I decreases TCP impacts in the underlying AF services, and helps the edge marking module to properly mark packets.So to speak, TMS_I can be not a marker used to mark packets in order to classify the service in the core of a network, but an indicator, also called *virtual source marker* that notifies TMS_II in the edge of a network of the marking rate. Second, the edge embedded marking module(TMS_II) monitors incoming traffic flows from users at the edge of the network against the profile that the users have agreed with the service provider. TMS_II measures the number of the marked packets (or marking information) from sources and re-marks aggregated flows for the profile that the users have agreed with the service provider. In this case, we elaborate a fairness strategy for sharing excess bandwidth of a bottleneck link. In [2], it showed that a source-integrated packet marking engine(PME) properly kept up marking rate rather than a source-transparent PME, because the measurement of throughputs against the reservation rate at the source is accomplished more exactly than at the edge of a network. We also assume that TMS_I properly marks packets at sources, and each host pours the packets into the edge of the network. Therefore, a fiducial point of fairness strategy is the marking rate(or marking information) of traffic flows from users. TMS_II re-marks(or marks) aggregated flows for total target rate of all the flows.

Note that in this case, sources' target rates are different from the edge's target rate. Let R_i denote the reservation rate of a flow i and C represent the capacity of a bottleneck link in the network. The target rate of each source is a reservation rate(R_i) of which user informs the edge, in order to be assured of bandwidth for delivering data. On the other hand, a flow's target rate(T_i) in the edge of a network is the reservation rate(R_i) plus variable value which is the marking rate to the excess bandwidth(E). The excess bandwidth at this bottleneck link is then given

$$E = C - R_i = C - \sum_{i=1}^{n} R_i \ . \tag{1}$$

Therefore, a flow's target rate in the edge of the network is given by

$$T_i = R_i + X_m \times E = R_i + X_{mi}(C - \sum R_i) \ . \tag{2}$$

Where, X_{mi} is proportional to the probability of marking from a source. That is, it is determined depending upon fraction between the marking rate of each flow and the average marking rate(4).
Development for X_{mi} is as follows: First, the probability of marking for each flow i is the number of the marked packets(N_{mi}) by throughput Thr_i, or In-profile window by congestion window.

$$m_i = \frac{N_{mi}}{Thr_i} = \frac{iwnd}{cwnd} \ . \tag{3}$$

Where, *iwnd* and *cwnd* represent window of the marked packets and congestion window, respectively. Next, the average marking rate $E[m_i]$ of all the flows at the edge of a network is

$$\begin{aligned} E[m_i] &= \frac{1}{n}\sum_{i=1}^{n} m_i \\ &= \frac{1}{n}\sum_{i}^{n}\left(\frac{iwnd}{cwnd}\right)_i \\ &= \frac{1}{n}\sum_{i}^{n}(Mark_p)_i \ . \end{aligned} \tag{4}$$

Where, $Mark_p$ also represents the probability of marking in TMS_I algorithm of the. Finally, X_{mi} is developed by using the fraction between (3) and (4), that is, X_{mi} represents a relative value against the datum point, $E[m_i]$. The fraction of X_{mi} varies from 0 to 1 depending upon m_i and $E[m_i]$. For simplicity, we assume that X_{mi} has uniform distribution, so coefficient 1/2 in (5) represents average value of uniform distribution.

$$X_{mi} = \frac{m_i}{2E[m_i]} = \frac{m_i}{2\frac{1}{n}\sum_i^n m_i} = \frac{nm_i}{2\sum_i^n mi} . \tag{5}$$

3.2 Suitable-Marking Strategy: Two Markers System-I Algorithm

In this subsection, we focus on the marking algorithm, so called the suitable-Marking strategy that is implemented in the virtual source marker(TMS_I). Congestion window(cwnd) consists of two windows in this algorithm[9]: *In-profile window(iwnd)* and *Out-of-profile window*(ownd). *In-profile window* represents the number of marked packets per-RTT needed to achieve the reservation rate that is guaranteed by a contract with Bandwidth Broker. O*ut-of-profile window* signifies the number of unmarked packets per-RTT that are outstanding. Each window has its respective threshold (issthresh and ossthresh).

Program code 1 and 2 illustrate the modified TCP congestion control algorithm. Some of the earlier work [2], [9] has focused on similar issues in networks where marking policies are different from the one studied here. Our study extends the earlier work and proposes new approaches to improving the realization of bandwidth guarantees.

In the starting point of this algorithm, how can we determine the probability of marking(*Mark*$_p$)? It is generally determined depending upon fraction between the measured and target throughputs. The window size increases or decreases according to throughput against the reservation rate. When either window is above its threshold, it increases linearly.

In Program code 2, whenever OUT packets loss occurs, O*ut-of-profile window(*ownd) reduces the half of its window size. On the other hand, the loss of IN packets indicates that congestion window reduce the half of its window size
A virtual source marking module(TMS_I) properly keeps up marking rate rather than an edge-embedded marking module. While the established edge marker, for example, is fairly effective in maintaining the observed throughput close to the reservation rate, it often marks more packets than required. A sender recognizes packets loss, and determines whether the lost packet was sent as a marked or an unmarked packet. The

loss of IN-marked packets represents an acute congestion situation in the network, and congestion window reduces the half of its window size. That is, if the loss of IN-marked packets occurs frequently, the congestion window size will be maintained low. A virtual source marking module, however, will decrease TCP impacts over AF services, because it properly regulates marking rate according to the condition of a network. Thus, since the probability that IN-marked packets are excessively dropped at the core of the network is reduced, congestion window will not be often decreased unlike the established edge marker.

The goals of the TMS_I provide modified congestion control that reduces TCP dynamics over AF services and a new datum point for fairly sharing the bottleneck bandwidth. We deal with Usage of the datum point of the edge-embedded marker in subsection 3.3. Therefore, the strategy providing the datum point alleviates somewhat of the problem of introducing unfairness within aggregated flows on *aggregation marking*.

Program code 1: TMS_I algorithm(cwnd open)

```
When new data is acknowledged
  iwnd = Mark_p * cwnd;
  ownd = cwnd - iwnd;
    if(Thr < Rr)
      if(iwnd < issthresh)
         iwnd += Mark_p;
      else
         iwnd += 1/2cwnd;
      if(ownd < ossthresh)
         ownd += (1- Mark_p);
      else
         ownd += 1/2cwnd;
    else
      if( 0 < iwnd)
         if(iwnd < issthresh)
            iwnd -= 1-Mark_p;
         else
            iwnd -= Markp;
      else        /* ownd = cwnd */
         if(ownd < ossthresh)
            ownd += 1;
         else
            ownd += 1/cwnd;
       cwnd = iwnd +ownd;
       Mark_p = iwnd/cwnd;
```

Program code 2: TMS_I algorithm(cwnd close)

```
When the third duplicate ACK in a row is
  iwnd = Mark_p * cwnd;
  ownd = cwnd - iwnd;
    if(out packet loss)
       ossthresh = ewnd/2;
```

```
      ownd = ossthresh;
   else
      cwnd = cwnd/2;
  issthresh = $Mark_p$ * cwnd;
  ossthresh = cwnd - issthresh;
```

3.3 Marking Rate-Based Fairness Strategy: Two Markers System-II Algorithm

The edge-embedded marker(TMS_II) elaborates a new marking strategy based on marking rates from sources in order to fairly share bottleneck bandwidth. In this case, the edge-embedded marker closely cooperates with sources to accomplish the fairness strategy, called “marking rate-based fairness”. Notice that fairness is defined as how well the throughput of all individual flows of aggregated traffic realizes their target rates(Ti).

We assume that an edge router receives information of the reservation rate which each user wants to be assured of, and it contracts with the service provider(or bandwidth broker) for bottleneck link bandwidth of the network. In this mechanism, the edge router sets the target rate different from the source: Recall the numerical expression (2) to mind. As stated in subsection 3.1, the target rate of each flow at the edge of a network includes excess bandwidth in proportion to the probability of marking from the sources, because the sources suitably mark packets against their target rates, that is reservation rates. Thus, the magnitude of marking rate represents increase or decrease in demand for bandwidth. If the number of marked packets, for example, exceeds the threshold value, that is the average marking rate, *E[m_i]*, the edge-embedded marker considers that the flow wants more bandwidth than others in order to achieve its reservation rate. Therefore, the flow is marked more and has a higher target rate than others.

Program code 3 illustrates the TMS_II algorithm that considers the “marking rate-based fairness” policy and a congestion situation.

Program code 3: TMS_II algorithm

```
  if($Th_t$ < C)
     For i -> i=1 to i =n
     Ti = $R_i$+($X_{mi}$ (C- $R_t$));
       if($Thr\_i$ < $T_i$)
          Mi += !|$T_i$ - $Thr\_i$|;
       else
          Mi -= !|$T_i$ - $Thr\_i$|;
   else if($Th_t$ == C)
      exit;
   else
      For i -> i = 1 to i = n
      $T_i$ = $R_i$
        if($Thr\_i$ < Ti)
           $M_i$ -= "|$T_i$ - Thr_i|;
        else
           $M_i$ -= #|$T_i$ - $Thr\_i$|;
```

```
/*  $M_i$: marking rate of ith flow
    $T_i$: target rate of ith flow,
    $R_i$: reservation rate of ith flow,
    Excess bandwidth = C - $R_{tr}$
    $X_m$ = (0.5 * $Mark_p$)/E[$m_i$]
    ! ≤ " ≤ #, ! s, ", and # is experimental values   */
```

First, TMS_II observes the total throughput of aggregated flows and compares it with the bottleneck link bandwidth(*C*) of the network. This comparison has two objectives: One is to consider a congestion situation, the other is to ensure the coherence of the suitable-marking strategy. The first and seventh lines in program code3 show that the edge-embedded marker fairly allocates the excess bandwidth of the bottleneck link. The eighth and ninth lines represent that the edge marking rate is equal to the marking probability of the sources(or marking information), and marking mechanism for a congestion situation appears from the tenth to the sixteenth lines. Then, if the total throughput doesn't reach the capacity *C*, it again measures whether the throughput of a flow i realizes the target rate (T_i), the crux of the fairness strategy, or not. Note that if the total throughput is more than the capacity *C*, the target rate of each flow becomes its reservation rate. Now, changes of the marking rates for each flow follow: According to a network situation, TMS_II applies different scaling to marking rate. That is to say, it uses ! when there is enough bandwidth to achieve the contract rate at bottleneck link of a network, and " or # when congestion occurs. We consider that " is bigger than ! and # is bigger than ". The idea behind difference is to quickly reduce loss of the IN-marked packets in oversubscribed state. When there is an acute congestion at the bottleneck link of a network and loss of the IN-marked packets occurs, the assured service isn't guaranteed due to TCP dynamics. Therefore, the differentiated scaling mechanism can have important meaning in oversubscribed network.

In short, sources inform an edge-embedded marker of suitable marking rate, then edge allocates fairly bandwidth to flows based on the marking rate of each source. Therefore, we can say that marking rate-based fairness strategy alleviates the problem of leading to an inefficient utilization of the reserved bandwidth on flow marking.

4 Simulation and Analysis

In this section, we present the simulation results with two marking algorithms we have described in the previous section. The simulation was done with the network simulator-2(ns-2). For the sake of simulation, we used a network with the configuration shown in Figure 2. In the simulation, we have 10 sources (1 through 10 counting downwards) that communicate with one of ten different destinations. We conducted two scenarios: oversubscribed and non-oversubscribed network. In the first scenario, the aggregate reservation rate is 30Mbps, and the bottleneck capacity is set to 40Mbps so that the bottleneck is not oversubscribed. In the second scenario, the aggregate reservation rate is 50Mps, and the bottleneck capacity is also set to 40Mbps so that the bottleneck link experiences congestion. We assume that the RTT without queuing delay of each flow is randomly pocked from 80 to 220 ms. The sources are all

TCP-Reno sources(unless specified otherwise). For the RIO implementation, the routers use RED with the values of 200 packets, 400packets, and 0.02 for *min_in, max_in*, and P_{max_in} and 50 packets, 100 packets and 0.5 for *min_out, max_out*, and P_{max_out}.

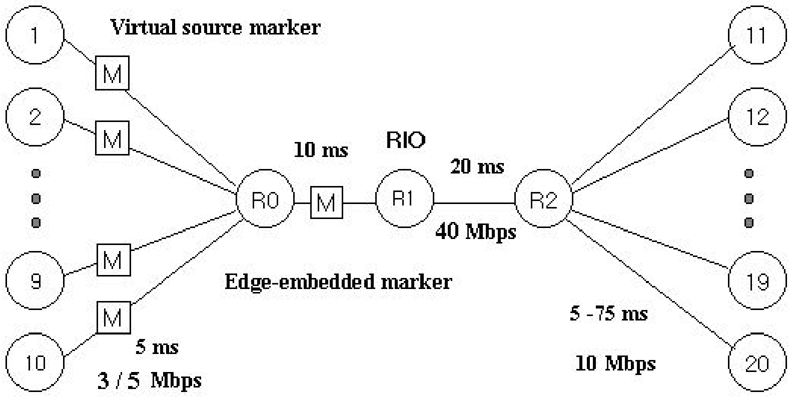
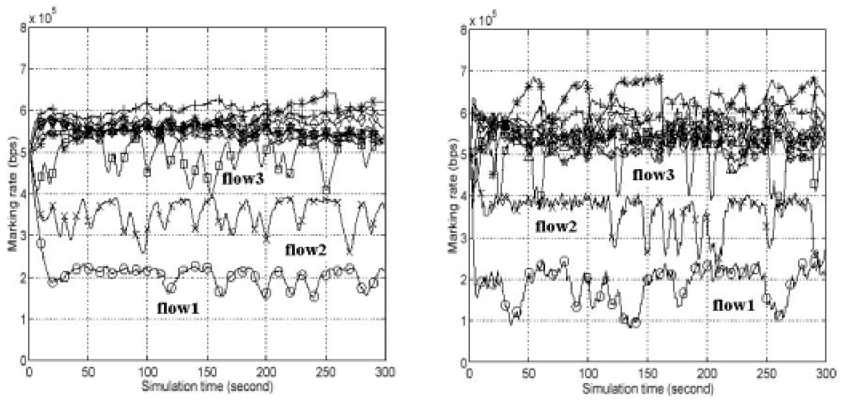


Fig. 2. Simulation topology

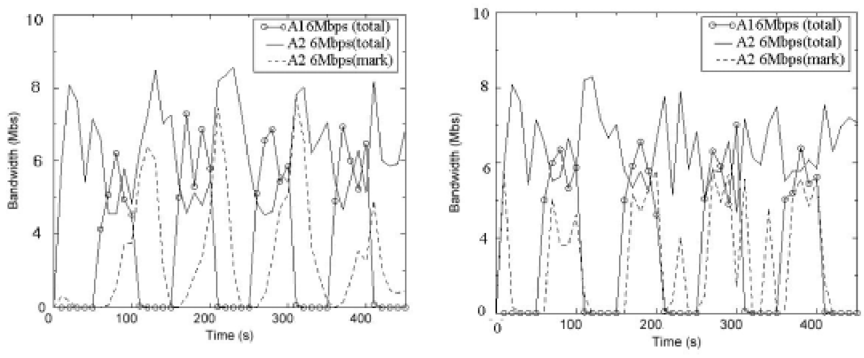
First of all, we investigate the simulation results with suitable-marking algorithm as the role of the virtual source marker. Since TCP windowing restricts the aggregated flows from competing for the excess bandwidth, the established edge marker continuously over-marks its packets as shown in Figure 3(a). It shows somewhat constant marking curves. However, since TMS_II determines marking rate based on the suitable-marking algorithm of sources, Figure 3(b) shows that the number of marked packets to accomplish the target rate is fewer than in the case of Figure 3(a). In the graphs below, flow 1 and flow2 demonstrate a striking contrast between *excess-marking* and *suitable-marking*. In runtime from 100(s) to 200(s), while flow 1 and flow 2 in the Figure 3(b) mark each packet of a flow complying with the changing condition of the network, those in the Figure 3 (a) do not so.

Second, we study the proper values of scaling factors !, ", and # through simulation, respectively. In Figure 4 (a), M_i is adjusted in steps such as $\alpha = 0.01$, $\beta = 0.05$, and $\gamma = 0.10$. As the figure shows, A2 is slow in reacting to changes in the network, and the marking rate lags behind the changes in the network load, slowly rising in response to an increased traffic load and slowly falling in response to a decreased traffic load. To investigate the other side, the simulation was repeated while allowing Mi to be updated in scaling factors of $\alpha = 1.0$, $\beta = 1.0$, and $\gamma = 1.5$. That is, when more bandwidth is needed to achieve the target rate, packets are marked more. Otherwise, packet marking is quickly turned off. Figure 4(b) shows the results from this simulation. As expected, M_i adapts very quickly to the changes in the network load, thus allowing the flow to achieve its target rate during periods of increased traffic load. This rapid response also allows the edge-embedded marker to turn off packet marking quickly when it detects that the available bandwidth is sufficient to satisfy the target rate.



a) The established marker b) Proposed TMS marker

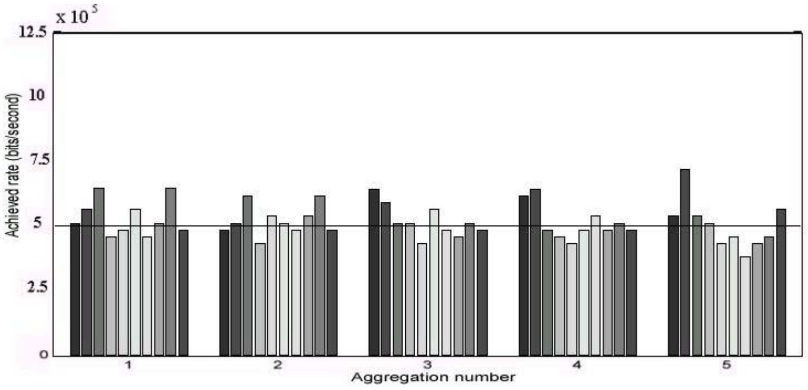
Fig. 3. Suitable-marking algorithm: When the established marker is applied, graph of a) represents somewhat constant marking rate. However, when suitable marking algorithm is applied, graph of b) represents variable marking rate according to a certain situation of the network. For example, flow 1 in graph of b) properly marks packets in comparison with graph of a)



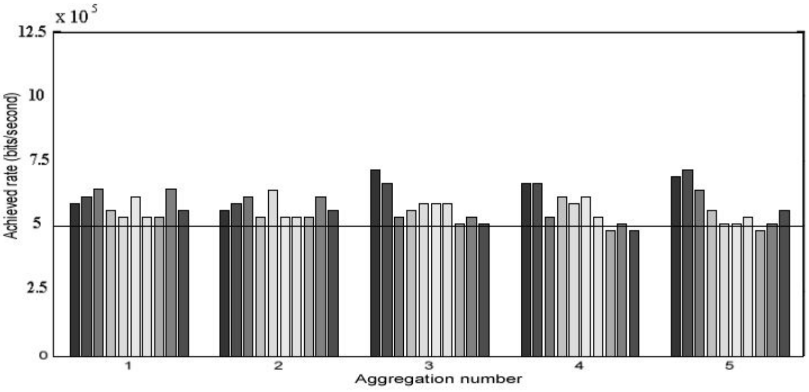
a) α = 0.01, β =0.05, and γ =0.10 b) α = 1.0, β = 1.0, and γ = 1.5

Fig. 4. Marking Packet through scaling factor:

Finally, we present the simulation results with marking rate-based fairness algorithm described in the previous section. We set that reservation rate of each flow is 5Mbps, and compare two marking schemes: One is the aggregation marking, the other is marking strategy of TMS. As stated above, the target rate of a flow i in TMS varies in proportion to the probability of marking from the sources. Figure 5 represents the results that the throughputs of all individual flows of aggregated traffic realize their target rates. Each flow in Figure 5 (a) often fails to achieve their target rates, because it receives bandwidth with no information about its state. That is to say, aggregation marking marks packets based on only aggregation state, without any knowledge about individual flows. However, in Figure 5 (b), each flow satisfies its reservation rate and shares the excess bandwidth of the bottleneck link according to the probability of his marking if the total throughput is less than the capacity of a network.



a) Aggregation marking strategy



b) Marking rate-based fairness strategy

Fig. 5. . Throughput for marking rate-based fairness

Two Markers System shows the improvement of performance about several parameters we used to simulate. However, an implementation of virtual source marking modules at source causes an issue of scalability in a differentiated services network through modifying TCP source, in addition, ! , ", and # should be set at their optimal values for the burstiness problems occurred owing to coarse scaling value.

5 Conclusion

The strongest point of the Two Markers System proposed in this paper is that two marking modules are able to properly mark packets and fairly share the bandwidth for the target rates: suitable-marking and marking rate-based fairness strategies(markers). First, TMS_I using the suitable-marking algorithm properly adjusts the probability of marking(or the information of marking) according to the changes of the network, and notifies TMS_II, called the edge-embedded marker, of the marking rates that will be used to mark aggregated flows at the edge of the diffserv network. Next, TMS_II at the edge of the network marks packets based on the marking rates of sources, and allocates flows the bottleneck bandwidth in view of proportional fairness, called *marking rate-based fairness* modified *flow aware aggregation marking* strategy. We have simulated a TMS model to study the effects of several factors on the throughput rates of TCP flows in a RIO-based Differentiated Services network. Despite the scalability and burstiness issues, the simulation results, as expected, showed that Two Markers System reduced TCP impacts over assured service and fairly shared the bottleneck bandwidth.

In the near future, we will find the optimal values of scaling factor, !, ", and #, in order to alleviate the burstiness problems of coarse scaling values, and study efficient scheme that informs edge-embedded marker of the information of marking.

Acknowledgement

This research was supported in part by Information and Telecommunication Research Center(ITRC) contracts. We are deeply grateful to Asian Pacific Advanced Network-Korea Consortium(APAN-KR) members for technical discussion on Differentiated Services related issues.

References

1. W. Feng, D. Kandlur, D. Saha, andt K. Shin: Adaptive Packet Marking for Providing Differentiated Services in the Internet, In Proc. ICNP'98, Austin, TX, Oct. (1998) 108-117
2. W. Feng, D. Kandlur, S. Saha, and K. Shin: Understanding TCP Dynamics in an Integrated Services Internet, in Proc. NOSSDAV'97, St. Louis, MO, May (1997). 279-290
3. S. Floyd and V. Jacobson, Like-sharing and Resource Management Models for Packet Networks, IEEE/ACM Transactions on Networking, vol. 3 no. 4, August (1995) 365-386
4. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski: Assured Forwarding PHB Group, RFC2597, June (1999)
5. K. Nichols, S. Blake, F. Baker, and D. Black: Definition of the Differentiated Service Field (DS Field) in the Ipv4 and Ipv6 Headers, RFC2474, Dec (1998)
6. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss: An Architecture for Differentiated Services, RFC2475, Dec. (1998)
7. A. Feroz, A. Rao, and S. Kalyanaraman, TCP-friendly traffic marker for IP differentiated services, in Proc. IWQoS'2000, Pittsburgh, PA, June (2000) 35-48

8. V. Jacobson and M. Karels: Congestion Avoidance and Control, in Proc. SIGCOMM'88, Stanford, CA, August (1998) 314-329
9. I. Yeom and A. L. N. Reddy: Realizing Throughput Guarantees in a Differentiated Service Network, in Proc. ICMCS'99, Florence, Italy, June (1999) 372-376
10. N. Seddigh, B. Nandy, P. Pieda: Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network, in Proc. GLOBECOM'99, Rio De Janeiro, Dec. (1999)
11. C. Gbaguidi, H. J. Einsiedler, P. Hurley, W. Almesberger, and J. Hubaux: A Survey of Differentiated Services Proposals for the Internet, SSC Tech. Report SSC/1998/020, http://sscwww.epfl.ch, May (1998)
12. J. Ibanez, K. Nichols: Preliminary Simulation Evaluation of an Assured Service, IETF Internet Draft, draft-ibanez-diffserv-assured-eval-00.txt, August (1998)
13. D. Clark and W. Fang: Explicit Allocation of Best Effort Packet Delivery Service, http://difserv.lcs.mit.edu/exp-alloc-ddc-wf.ps (1997)

Aguri: An Aggregation-Based Traffic Profiler

Kenjiro Cho[1], Ryo Kaizaki[2], and Akira Kato[3]

[1] Sony Computer Science Labs, Inc., Tokyo 1410022, Japan
kjc@csl.sony.co.jp
[2] Keio University, Fujisawa 2528520, Japan
kaizaki@sfc.wide.ad.jp
[3] The University of Tokyo, Tokyo 1138658, Japan
kato@wide.ad.jp

Abstract. Aguri is an aggregation-based traffic profiler targeted for near real-time, long-term, and wide-area traffic monitoring. Aguri adapts itself to spatial traffic distribution by aggregating small volume flows into aggregates, and achieves temporal aggregation by creating a summary of summaries applying the same algorithm to its outputs. A set of scripts are used for archiving and visualizing summaries in different time scales. Aguri does not need a predefined rule set and is capable of detecting an unexpected increase of unknown protocols or DoS attacks, which considerably simplifies the task of network monitoring.
Once aggregates are identified and profiled, it becomes possible to make use of the profile records to control the aggregates in best-effort traffic. As a possible solution, we propose a technique to preferentially drop packets from aggregates whose volume is more than the fairshare. Our prototype implementation demonstrates its ability to protect the network from DoS attacks and to provide rough fairness among aggregates.

1 Introduction

Traffic monitoring is essential to network operation in order to understand usage of the network and identify abnormal conditions or threatening activities. Also, longer-term monitoring is needed for capacity planning or for tracking trends. Flow-based traffic profiling in which packets are categorized into traffic types and usage information is recorded for each type is commonly used for traffic monitoring [3,9]. Flow-based traffic monitoring, combined with visualization techniques, provides a powerful tool to understand network conditions [2,16,20,21].

However, a weakness common to the existing flow-based monitoring tools is that, to identify traffic types, predefined filter rules are needed. Filter rules are used to classify packets by examining fields in the packet header. Thus, without *a priori* definitions of traffic types, packets cannot be identified. Flow-based monitoring is facing a difficulty identifying new protocols and dynamically assigned ports. Even for known traffic types, it is not practical to list all possible combinations in the rule set so that minor traffic types are often left undefined and remain unidentified.

On the other hand, the current Internet is exposed to the menace of Denial of Service (DoS) attacks, and DoS attack detection is the highest priority for network operation. The rule-based approach lacks an ability to detect DoS attacks since forged packets can have arbitrary traffic types.

We have been monitoring the WIDE research backbone for years [8], and badly in need of an adaptive monitoring tool for trouble detection, usage reporting and long-term trend analysis. Our focus is traffic measurement to aid network operation, and thus, concise and timely summary reports are more important than precise and detailed reports.

To this end, we have developed a software package called aguri. Aguri uses a traffic profiling technique in which records are maintained in a prefix-based tree and a compact summary is produced by aggregating entries.

Powerful is the feature to produce a summary of summaries applying the same algorithm to its own outputs. Thus, derivative summaries can be produced in different time scales desirable for a specific monitoring purpose. A set of scripts have been developed to visualize summaries. It is also possible to extend the profiler as a protective measure against DoS attacks.

Aguri is targeted for near real-time, long-term, and wide-area traffic monitoring. Because automatic aggregation is used for profiling, our approach provides rough usage reports which may not be precise so that it is complementary to the existing tools.

2 Overview of Aguri

The core idea of an aggregation-based profiling is to aggregate flow entities for profiling. Small volume flows are aggregated until the volume of the aggregate becomes large enough to be identified. A summary output reports the profile of aggregates. An entry in an address profile can be a single host if it consumes a certain portion of the total traffic, or an aggregate if each host entry is small but the aggregate becomes non-negligible. Thus, a limited number of entries are produced, yet it never fails to report high volume entries.

Figure 1 illustrates the concept. A tree before aggregation is on the left and the corresponding tree after aggregation is on the right. Each node in the tree shows the address space represented by an address prefix and its prefix length. A leaf node corresponds to a single address. The size of a node shows the traffic volume of the node. The usage information recorded at leaf nodes can

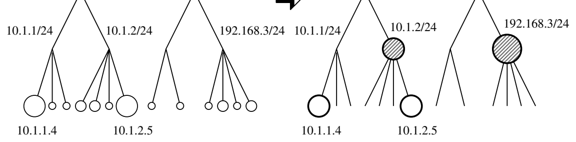


Fig. 1. Aggregation profiler concept: small entries are aggregated into aggregates

be aggregated to the shaded internal nodes in the right tree, and a summary reports only the remaining nodes in the right tree.

Summary Profile. It is important to produce concise summary profiles. When a traffic profile is too detailed, important symptoms are buried in excessive data, and often left unnoticed. Each summary profile produced by aguri is compact since small entries are aggregated in a profile.

Aguri produces four separate profiles for source addresses, destination addresses, source protocols and destination protocols. IP addresses are designed to be hierarchical and aggregatable so that it is natural to apply aggregation. Both IPv4 and IPv6 are supported in address profiles. Although protocol numbers are not hierarchical, the same technique can be used to identify port ranges. We concatenate the IP version, the protocol number and the TCP/UDP port number to create a 32-bit key for a protocol profile. A summary reports the total byte count used by each aggregate.

The four separated profiles are effective to capture hostile activities. A victim of a distributed DoS attack will be easily identified in the destination address profile. An originator of port scanning will be identified in the source address profile. A random attack will be identified as a range of addresses as long as some locality exists for the targets. If the locality is unusually low, it is another symptom of a random attack.

Spatial Aggregation. The basic algorithm of the spatial aggregation is quite simple. If there is no resource constraints such as memory consumption or execution time, we could profile every address and protocol occurrence in every packet and, at the end, aggregate entries whose counter value is less than an aggregation threshold. This approach would be acceptable for post-analysis of a saved packet trace. For near real-time monitoring, however, we approximate the above algorithm in exchange for the precision, by managing a fixed number of nodes in the tree using a variant of the Least-Recently-Used (LRU) replacement policy.

When a leaf node is reclaimed, the counter value of the node is aggregated to its parent node. The advantage of this approach is that counter values are never lost even though the resolution is reduced.

To produce a summary output, aguri walks through the tree in the post-order and aggregates nodes if the counter value of a node is less than the aggregation threshold, or outputs the node information if the counter value is above the threshold.

To continue profiling, it is enough to reset the counter of each node; the current tree and the LRU list are kept in tact as a cache, and used for the next profiling period.

Temporal Aggregation. The same algorithm can be used to produce a summary of summaries. Aguri can read its summary outputs, reaggregate them, and produce a new coarse-grained summary. For instance, a 1-hour-long summary can be created out of 60 1-minute-long summaries.

In this paper, an "initial summary" is used to represent a summary directly produced from non-aggregated sources such as captured packets. A "derivative summary" represents a summary produced from summaries.

The method is suitable for archiving profiles since a summary can be created in different time scales from a set of archived summaries. It is also possible to control the resolution by changing the aggregation threshold. The process to generate and archive derivative summaries can be easily automated. Network operators will usually look at only coarse grained summaries but can look into fine grained summaries if necessary.

Archiving and Visualization Utilities. A summary output is in a plain text format so that it is easily processed by various scripts. For archiving, a script is periodically invoked to generate and archive derivative summaries in different time scales such as hourly, daily, monthly, and yearly summaries. The size of a summary is about 5KB so that a small amount of disk space is required for archiving summaries.

Text-based summaries can be converted to a variety of visual images. We have developed a set of scripts for visualization to aid operators to find unusual conditions in summary outputs.

Application for Traffic Control. Once aggregates are identified and profiled, the profile records can be used for traffic control. There are many possible approaches to control aggregates. For example, a rate-limiter can be installed at a firewall to protect the network from a high-bandwidth aggregate [17].

We propose an aguri three color marker (aguriTCM) that combines an aggregation-based profiler with a preferential packet dropping mechanism. The aguriTCM identifies aggregates whose traffic volume is more than the fairshare, and probabilistically raises the drop precedence for those aggregates. The aguriTCM provides rough traffic management based on aggregates in best-effort traffic; the resolution of the control is limited by the resolution of an aggregate in the profile.

Our approach uses Diffserv components as building blocks but the primary target is a stand-alone protection mechanism to minimize the effect of DDoS or flash crowd in best-effort traffic. It also provides rough fairness among aggregates.

3 Related Work

MRTG [20] and its successor RRDtool [19] create time-series round-robin databases. They store numerical time-series data and automatically aggregate it into averages over time. Our idea of producing a summary from summaries is inspired by MRTG and RRDtool but differs in combining temporal aggregation with spatial aggregation.

Traditional flow-based monitoring tools such as NeTraMet [2] and FlowScan [21] require predefined rules to monitor a specific type of traffic. For example, in order to monitor HTTP traffic, they need to be instructed to identify TCP port 80. The approach with explicit and fixed rules has limitations on identifiable traffic types. Especially, it is a problem to cope with unknown protocols or DoS attacks.

Another approach is to report the top N flows by sorting the flow list [24,4]. Although it does not need a rule set, there could be limitations on the maintainable number of flows or a flooding attack could easily overflow the list. Hence, it is not suitable for detecting DoS attacks. In our approach, a flooding attack may be able to reduce the resolution of the profile but the counter values are never lost. It is resilient to DoS attacks in addition to requiring no rules.

Dynamic identification of a flow is also addressed in the context of congestion control and DoS prevention. Floyd *et al.* in [11] argue on the need for end-to-end congestion control, and further, on the need for mechanisms in the network to detect and restrict unresponsive or high-bandwidth best-effort flows in times of congestion. They suggest to use the RED drop history as samples to identify misbehaving flows. The concept is known as a RED penalty-box [6].

This idea is further extended and detailed in order to cope with DDoS attacks and flash crowds [17]. It consists of a mechanism to identify aggregates, a local rate-limiter mechanism, and a pushback mechanism to propagate protective actions to neighbors. The proposed technique to identify high-bandwidth aggregates is based on the destination address in the drop history, and clusters the addresses into aggregates. The approach of identifying high-bandwidth aggregates and regulate them is similar to ours in the concept.

While their focus is to identify misbehaving flows, our focus is a traffic profiler which monitors and reports the network not only under congestion but all the time. Our observation is that a network point needing a protection mechanism is often a point to be monitored. Hence, it is practical to provide a combined solution both for performance and for simplicity. The combined method comes with visible monitoring outputs so that it could be advantageous to deployment.

4 Implementation

Aguri, as shown in Figure 2, is implemented as a user program on UNIX. The input modules on the left translate different input formats into a 4-tuple (tree, key, prefix-length, count) and pass them to the profiler engine in the center. Aguri prints summaries to the standard output or a file.

The first input module reads aguri's summary outputs from files or from the standard input to produce a derivative summary. The second input module is an interface to the pcap library [15] that captures packets from a live network

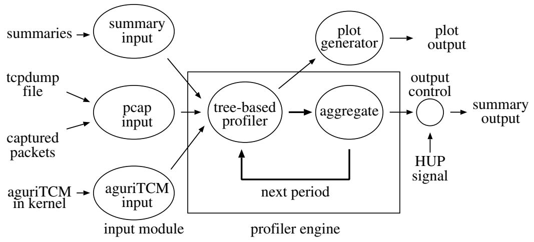


Fig. 2. Aggregation profiler implementation model

```
%!AGURI-1.0
%%StartTime: Sat Jan 06 14:00:00 2001 (2001/01/06 14:00:00)
%%EndTime:   Sat Jan 06 14:00:05 2001 (2001/01/06 14:00:05)
%AvgRate: 17.05Mbps

[dst address] 10658367 (100.00%)
0.0.0.0/0       105652 (0.99%/100.00%)
 0.0.0.0/2      196398 (1.84%/1.84%)
 128.0.0.0/1    141492 (1.33%/97.17%)
        133.28.0.0/16   146217 (1.37%/11.08%)
                133.28.21.21    179320 (1.68%)
         133.28.128.0/17        257220 (2.41%/8.03%)
                133.28.128.14   127541 (1.20%)
                133.28.202.127  470854 (4.42%)
   152.0.0.0/5  157159 (1.47%/25.69%)
        152.10.0.0/16   336636 (3.16%/20.28%)
         152.10.0.0/17  433037 (4.06%/15.16%)
                152.10.1.247    1182481 (11.09%)
                152.10.135.189  208992 (1.96%)
      156.96.0.0/11     253884 (2.38%/3.94%)
        156.114.0.0/16  165979 (1.56%/1.56%)
   168.0.0.0/5  315417 (2.96%/47.96%)
                168.89.12.93    275740 (2.59%)
      173.96.0.0/12     465797 (4.37%/42.42%)
          173.106.176.0/20      248236 (2.33%/38.05%)
                173.106.177.162 440466 (4.13%)
                173.106.177.163 550897 (5.17%)
                173.106.177.172 602230 (5.65%)
                173.106.177.173 1498198 (14.06%)
                173.106.187.134 559784 (5.25%)
                173.106.187.135 155322 (1.46%)
   192.0.0.0/5  111918 (1.05%/8.45%)
    194.0.0.0/7 375630 (3.52%/7.40%)
                194.105.251.45  168327 (1.58%)
                195.130.218.237 244270 (2.29%)
  208.0.0.0/4   283273 (2.66%/2.66%)
%LRU hits: 82.62% (14511/17564)
```

Fig. 3. A sample output of a destination address profile

or reads a packet trace file saved by tcpdump [14]. The pcap interface allows us to evaluate our prototype using various tcpdump trace files. The third input module reads binary profiles produced by the aguriTCM in the kernel.

The profiler engine consists of the tree-based profiler and the aggregation module. The tree-based profiler accepts 4-tuples from one of the input modules, and maintains profile records in the trees. At the end of a profiling period, the aggregation module is called to produce a summary. While the aggregation module is walking through the tree in the post-order, each node is either aggregated or reported. To continue profiling, the profiler engine repeats this cycle.

4.1 Summary Output

Figure 3 shows an example of aguri's summary output. A summary starts with a header block, followed by a body block. Lines start with % are comment lines. The body block contains 4 profile types by default but only the destination address profile is shown in the figure. [1]

[1] IP addresses appearing in this paper are scrambled for privacy.

In the address profile, each row shows an address entry and is indented by the prefix length. The first column shows the address and the prefix length of the entry. When the prefix length is the full length, it is omitted in the output. The second column shows the cumulative byte count. The third column shows the percentages of the entry and its subtree.

The input for this example is a 5-second-long packet trace taken from a trans-pacific link of the WIDE backbone. The parameters of aguri is configured with 256 nodes and 1% aggregation threshold. Among 17,564 observed addresses, only 14 addresses are identified as individual addresses. 38.05% of the traffic belongs to 173.106.176/20; within this address space, 6 distinct addresses are identified. The number of individual addresses found in a typical summary is from 5 to 20. In our trans-pacific profiles, several individual addresses are still identified even in monthly summaries.

A source address profile looks similar. A source address profile tends to identify popular www or ftp servers, whereas a destination address profile tends to identify proxy servers and mirror servers.

Figure 4 shows source and destination protocol profiles. The first column shows a 32-bit key concatenating the IP version number (8bits), the protocol number (8bits), and the TCP/UDP port number (16 bits). For example, "4:6:80" represents IPv4/TCP/HTTP.

In this summary, 96.15% of the total traffic is TCP. Only four individual ports, TCP port 20 (ftp-data), 80 (http), 6346 (gnutella), UDP port 53 (dns), are identified in the source address profile. Note that the use of gnutella is automatically detected without any knowledge about gnutella's use of port 6346.

The destination protocol profile includes a larger number of dynamically assigned ports which are usually aggregated and shown as port ranges. A source protocol profile tends to identify protocols used by servers, and a destination protocol profile tends to identify clients.

4.2 Spatial Aggregation

The profiler engine implements the prefix-based aggregation algorithm. To produce summaries continuously in near real-time, we need an efficient algorithm in terms of CPU power and memory usage. An approximation limits the number of entries used in a tree, and thus, will make more aggregation than the ideal algorithm. As a result, it introduces two types of errors: (1) part of the counter value could be aggregated to the ancestors, and (2) the entry of a node close to the aggregation threshold could be removed and may not show up in the summary. These errors lower the precision but the impact would be limited. After all, these errors are unavoidable for derivative summaries since aggregation discards details. However, if an entry consumes a non-negligible volume of the total traffic, any approximation will be able to detect it.

To limit memory use and search time with variable length keys, we employ a Patricia tree. Patricia has been employed in the BSD kernel for the internal representation of the routing table [23], and its performance characteristics are

```
[ip:proto:srcport] 10570555 (100.00%)
0/0:0:0 4967 (0.05%/100.00%)
4:0/3:0 290382 (2.75%/99.95%)
4:6:0/0 164255 (1.55%/96.15%)
   4:6:0/3      540369 (5.11%/93.38%)
                4:6:20  663178 (6.27%)
                4:6:80  7329218 (69.34%)
        4:6:1024/8      106427 (1.01%/1.01%)
        4:6:1280/8      139741 (1.32%/2.75%)
         4:6:1280/9     150514 (1.42%/1.42%)
       4:6:1536/7       182444 (1.73%/1.73%)
     4:6:2048/5 564594 (5.34%/5.34%)
                4:6:6346        194004 (1.84%)
 4:6:32768/1    128925 (1.22%/1.22%)
                4:17:53 111537 (1.06%)
%LRU hits: 60.80% (10644/17506)

[ip:proto:dstport] 10570555 (100.00%)
0/0:0:0 4967 (0.05%/100.00%)
4:0/3:0 401919 (3.80%/99.95%)
4:6:0/0 579078 (5.48%/96.15%)
         4:6:0/9        327066 (3.09%/4.54%)
                4:6:80  152813 (1.45%)
       4:6:1024/7       419016 (3.96%/17.12%)
         4:6:1024/9     781275 (7.39%/7.39%)
        4:6:1280/8      609679 (5.77%/5.77%)
       4:6:1536/7       597213 (5.65%/12.77%)
        4:6:1536/8      752782 (7.12%/7.12%)
      4:6:2048/6        666539 (6.31%/21.84%)
       4:6:2048/7       155545 (1.47%/15.54%)
         4:6:2176/9     387335 (3.66%/7.96%)
          4:6:2176/10   454168 (4.30%/4.30%)
        4:6:2304/8      645406 (6.11%/6.11%)
      4:6:3072/6        893343 (8.45%/8.45%)
    4:6:4096/4  172569 (1.63%/9.51%)
       4:6:4608/7       688892 (6.52%/6.52%)
                4:6:6346        143558 (1.36%)
  4:6:49152/2   492936 (4.66%/16.44%)
                4:6:49249       1107484 (10.48%)
                4:6:49635       136972 (1.30%)
%LRU hits: 53.96% (9446/17506)
```

Fig. 4. A sample output of protocols and ports

well understood. It is suitable to handle 32-bit IPv4 addresses and 128-bit IPv6 addresses.

Patricia is a full binary radix tree. All internal nodes have exactly two children so that when the number of leaf nodes is N, the number of internal nodes is $(N-1)$. Thus, it is suitable for use with a fixed number of nodes, and nodes can be preallocated.

Each node has a prefix as a key associated with its prefix length. The key of an internal node is the common prefix of its two children.

Our use of Patricia is different from the routing table. While the routing table lookup requires best-match, we have only exact-match. In our scheme, a new node is always created when no matching node is found. If there is no available free node, an old node is reclaimed to keep the number of nodes in the tree. Thus, node insertions and deletions occur during a lookup operation.

To update an entry record, the profiler looks up the entry in the tree, and updates the counter value of the entry. A lookup starts from the root node to a leaf node, checking prefix-matching. If the prefix matches with the internal node, the bit at $(prefixlen + 1)$ of the search key indicates which branch to follow; if the bit value is 0, take the left branch, otherwise, take the right branch. If the matching leaf node is found, the search terminates and the counter of the node is updated.

If the prefix does not match, it indicates no matching node exists in the tree. A new node is created and inserted into the tree. The key is assigned to the new node, and the count is set to the counter. An insertion always creates a leaf and a branch point since single branching is not allowed. The new branch point is inserted as a parent of the unmatching node; the other child of the branch point is the newly created leaf node. The common prefix of the two child nodes is assigned to the branch point. Similarly, deleting a leaf node removes the leaf and its parent. When deleting a node, the counter value is aggregated to its parent.

A fixed number of nodes are preallocated for a tree, and a variant of the LRU replacement policy is used for managing leaf nodes. If the number of nodes is 256, the tree has 128 leaf nodes since $(N-1)$ internal nodes are needed for N leaf nodes. The LRU is selected because it is simple, cheap and well-understood. The precision could be further improved by using an elaborate algorithm such as the frequency-based replacement [22] but there is a tradeoff between the precision and the efficiency. As already mentioned, the precision is not so important in our scheme and it is evaluated in Section 5.

Since the LRU reclaims a node even when its counter value is very large, a simple heuristic is added not to reclaim a node if the sum of the counter values of the node and its parent is larger than a threshold. The current reclaim exemption threshold is set to 3.123% or 1/32 of the total count.

In the middle of a profiling period, a snapshot of the tree contains nodes with small count values. Nodes whose count value is less than the aggregation threshold are aggregated at the end of the profiling period. The aggregation threshold is set to 1% of the total count by default. The profiler walks through the tree in the post-order so that aggregation and summary output can be done in one pass.

To continue profiling, the profiler just resets the counters and keeps the tree and the LRU list in tact as a cache for the next profiling period. The profiler could reset the counters when aggregating the nodes. However, a two-pass method is used in the current implementation to show the sum of the subtree for readability. The aguriTCM, on the other hand, omits the subtree sum and employs a one-pass method.

IPv4 and IPv6 addresses have different key length. They could be managed in a single tree but separate trees are currently used for ease of debugging. The aggregation threshold is computed from the combined total count so that there is no difference in the summary. On the other hand, the key length is the same for protocol trees so that the profiler uses merged trees.

The profiler uses the same algorithm to produce derivative summaries but there are subtle differences. The size of input sets is much smaller and there are less constrains on execution time or resource usage. Another difference in the

Patricia algorithm is that internal nodes are added to insert aggregates, while only leaf nodes are added for initial summaries. A single implementation is currently used for both initial and derivative summaries to reduce the maintenance cost but it could be separately optimized.

4.3 Archiving and Visualization Utilities

Archiving. Aguri prints summaries to the standard output or a file. On receiving a HUP signal, the output file is reopened so that the output file can be redirected to a new file. To archive summaries, a script is periodically invoked by *cron*. The script saves the current output file and sends a HUP signal to the running aguri program to switch the output file.

In our current setting, aguri produces a new summary every 5-seconds. A new summary file containing 24 summaries is created every 2-minutes. The script also generates hourly/daily/monthly/yearly summaries when crossing the time boundaries. It is also possible to customize the script to detect a certain condition and send an alert to the operator.

A summary output size varies depending on the traffic but is usually about 5KB. Uncompressed derivative summaries take about 150KB/hour, 3.5MB/day, 105MB/month and 1.2GB/year. If the initial summaries created every 5-seconds are saved, they consume additional 100KB for every 2 minutes. The initial summaries will take about 3MB/hour, 70MB/day, 2GB/month, and 24GB/year but these detailed summaries can be discarded after a certain period.

Plot Graph. Aguri supports a plot format output suitable to draw a plot graph. The plot format lists the counter values of the entries in a line; each line corresponds to a profiling period. It also supports conversion from byte-count to bits-per-second. A plot output is usually created from archived summaries and does not need to do in real-time. It is also needed to specify the number of entries in a plot. Thus, the plot generator uses a 2-phase algorithm which reads input files twice.

The first phase computes the cumulative byte count for each entry. At the end of the first phase, a sorted plot list is created, and the smallest entry is repeatedly aggregated until the number of nodes is reduced to the specified number. The second phase produces a plot format output for each period. For each period, if a node is not found in the plot list, it is aggregated to the nearest ancestor listed in the plot list. Hence, all counts are reflected to the plot.

Figure 5, 6 and 7 show examples of plot graphs taken from the trans-pacific link. The legend below the graph shows entries in the plot. Figure 5 plots destination addresses for 1 day on April 12, 2001, created from 2-minute summaries. Two individual addresses (148.65.7.36 and 167.215.33.42) are listed but there is no prominent address in terms of the bandwidth share.

Figure 6 plots source protocols for 10 days, from April 10 to 19, 2001, created from 1-hour summaries. The graph captures daily fluctuations of the total traffic and the high ratio of HTTP. In Figure 6, there is a change in the daily traffic

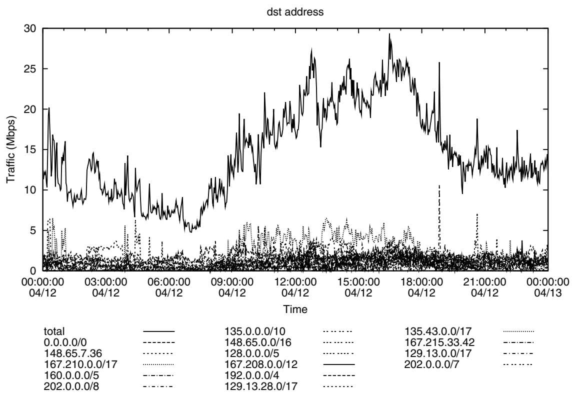


Fig. 5. A graph plotting 1-day destination addresses

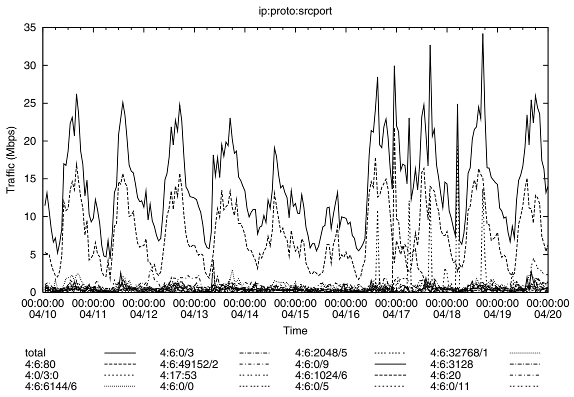


Fig. 6. A graph plotting 10-day source protocols

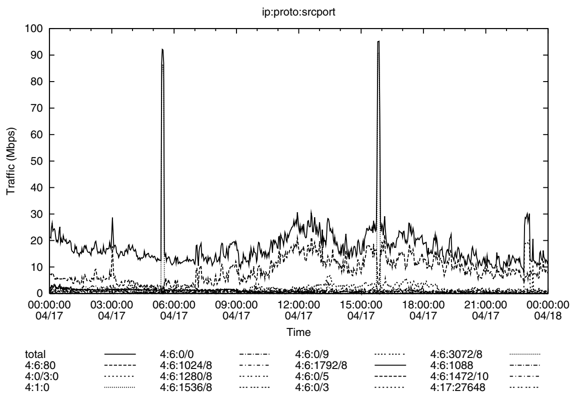


Fig. 7. A graph zooming into April 17th

pattern on the 17th. By zooming into the 17th as shown in Figure 7, we can see unusual surges of ICMP. It is a *smurf* attack and this is the cause of the distortion in the daily traffic. We can identify the target address and the address range of the originators by looking into the corresponding address profiles. This illustrates how plot graphs in different time scales can be used for trouble shooting.

Traffic Density Graph. Another graph format shows traffic density within the entire address space. From a summary, we can compute the traffic density in the address range of each aggregate, and create a time-series color graph. In a traffic density graph, the degree of traffic concentration is shown by colors and a change in traffic pattern is easily identified.

5 Evaluation

We have done a trace-driven evaluation using two 1-hour-long packet traces from the WIDE backbone [8]. Trace #1 is taken from a trans-pacific link, and trace #2 is taken from a link connected to a domestic IX. A set of shorter packet traces are extracted from the two traces. Table 1 shows the number of packets, the number of distinct addresses, and the observed rate in the traces.

The test configuration uses 256 nodes in a tree, 1% aggregation threshold, and 1/32 aggregation exemption threshold, unless otherwise specified.

5.1 Aggregation Accuracy

In our algorithm, the resolution of aggregation depends on the aggregation threshold. The number of nodes used in a tree, the replacement policy, the generation of derivative aggregation also affect the precision of a result.

Although accuracy is not the most important factor to the algorithm, it is better to understand the impact to the results. To measure the distortion in the resulting tree, we introduce the distortion index that provides a quantitative difference of two trees.

Table 1. packet traces used for evaluation

trace	length	# of packets	# of addresses	rate (bps)
#1	1sec	3929	775	20.92M
	5sec	19977	1884	21.12M
	60sec	242187	7297	22.44M
	3600sec	16351933	75530	25.55M
#2	1sec	1380	295	4.27M
	5sec	6664	786	3.72M
	60sec	113680	3617	7.10M
	3600sec	5289374	25981	3.91M

Distortion Index. The approximation in our algorithm introduces excessive aggregation in the resulting tree. We need to measure errors caused by the excessive aggregation, by comparing the resulting tree with the ideal tree. Traditional tree matching methods in graph theory (e.g., edit-distance) are not suitable for this purpose since they do not take aggregation into consideration.

Aggregation moves the counter value of a node to its ancestors but it never affects the other nodes. The aggregated value could be spread over multiple ancestors. Thus, we should do subtree-by-subtree comparison rather than node-by-node comparison.

Figure 8 illustrates the distortion. $T_1[k]$ on the left is a subtree at kth node in ideal tree T_1 and $T_2[k]$ on the left is the corresponding subtree in approximation T_2. The shaded portion of node i is aggregated to node k in the right subtree. When we compare $T_1[i]$ with $T_2[i]$, the volume of the shaded area is considered shifted by the distance from i to k that is the difference of their prefix length. $T_1[k]$ and $T_2[k]$ is considered equal since their subtrees have the same volume. If T_2 does not have a corresponding node, we assume a virtual node with size 0.

We introduce a distortion index to quantify the difference. Let $D_{12}[i]$ be the distortion index from $T_1[i]$ to $T_2[i]$. We compare the total count of the subtree at node i. $s_1[i]$ and $s_2[i]$ are the sum of the counters in $T_1[i]$ and $T_2[i]$, respectively. If $s_1[i]$ is larger than $s_2[i]$, the difference is considered to be aggregated into the ancestor nodes in T_2. Thus, we find the nearest ancestor k where

$$\frac{|s_1[k] - s_2[k]|}{s_1[k]} < \varepsilon$$

ε is an error term to allow small differences in size matching. We use 0.05 for ε. $d_{12}[i]$ represents the distance from i to k, normalized to the full prefix length.

$$d_{12}[i] = \frac{prefixlen(i) - prefixlen(k)}{prefixlen_{max}}$$

$r_{12}[i]$ is the ratio of the difference in the subtree coverage at node i, normalized to the subtree size.

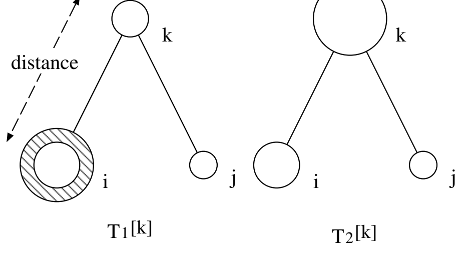


Fig. 8. distortion of two subtrees: the ideal tree on the left and the approximation on the right

$$r_{12}[i] = \begin{cases} \frac{s_1[i]-s_2[i]}{s_1[i]} & \text{if } (s_1[i] > s_2[i]) \\ 0 & \text{otherwise} \end{cases}$$

$w[i]$ is the weight of node i in the tree, and computed as the byte count of the node divided by the total byte count of the tree. Then, we get the normalized distortion at node i as

$$D_{12}[i] = w[i] \cdot r_{12}[i] \cdot d_{12}[i]$$

Each item ranges from 0 to 1.0. A small exponent, b, is added to each item as a bias towards small errors because small errors are expected by aggregation. We use 1.2 for b. The distortion index for the entire tree can be obtained as the sum of the indices. By making it symmetric, the distortion index becomes

$$D = \frac{\sum_{i \in T_1} w^b \cdot {r_{12}}^b \cdot {d_{12}}^b + \sum_{j \in T_2} w^b \cdot {r_{21}}^b \cdot {d_{21}}^b}{2}$$

This index, albeit not perfect, at least allows us to quantify the results. When two trees are exactly the same, D becomes 0. When one tree has all the count at leaf nodes and the other tree has all the count at the root node, D becomes 0.5. When there is no overlap, D becomes 1.0. For example, one tree has all the count at leaf nodes in the left branch and the other tree has all the count at leaf nodes in the right branch.

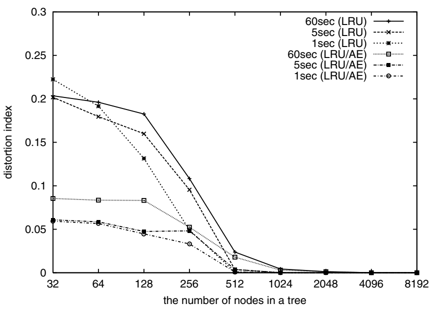
Accuracy Results. We use the distortion index to evaluate our LRU-based algorithm. Figure 9 shows the effects of the number of nodes and the profiling period length, with or without the heuristic added to the LRU algorithm, in the source and destination address trees of the two traces.

In the figure, "LRU" shows the simple LRU algorithm, and "LRU/AE" shows the LRU with the aggregation exemption threshold. The distortion index is computed with the ideal results in which there is no restriction on the number of nodes.

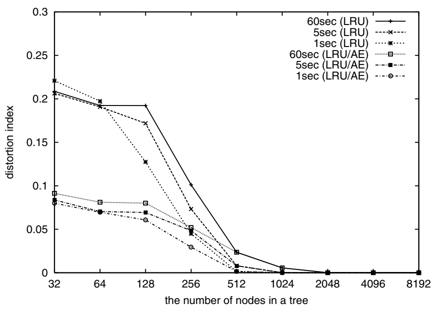
The effect of the different period length are tested by the traces with different length. Even though the number of the included addresses differs in orders of magnitude, the results look similar. It suggests that there is a locality in address occurrence, and thus, the results are not affected much by the trace length.

As expected, the simple LRU works well when there are enough nodes but the distortion becomes larger when nodes are insufficient. The aggregation exemption reduces distortion, especially when the profiler runs out of nodes.

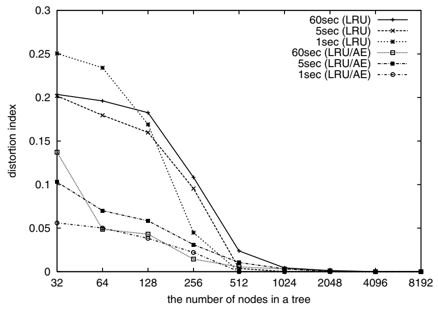
Table 2 shows the differences in summary generations. "3600s" shows the initial summary directly produced from the packet trace. This is the base summary for comparison. "1sx3600" is a second-generation summary produced from 3600 1-second summaries. "60sx60" is another second-generation summary produced from 60 60-second summaries. "1sx60x60" is a third-generation summary. 1-second summaries are first aggregated to 60 60-second summaries, and then, the final summary is created. "5sx24x30" is another third-generation summary.



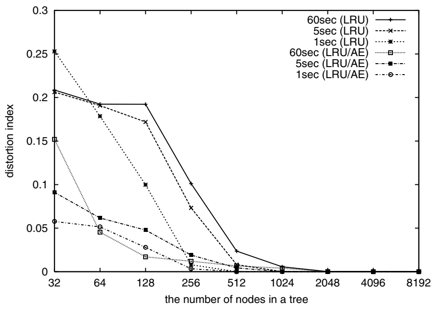
(a) trace #1 source addresses



(b) trace #1 destination addresses



(c) trace #2 source addresses



(d) trace #2 destination addresses

Fig. 9. distortion caused by LRU with varying tree size and the profiling period length

5-second summaries are first aggregated to 30 120-second summaries, and then, the final summary is created. The results show that the distortion introduced by summary generations is fairly small, which justifies our approach to create derivative summaries for temporal aggregation.

5.2 Performance

For every packet, aguri looks up the matching entry in the 4 trees and manages the LRU lists. When the number of nodes in a tree is N, the lookup operation runs in $O(\lg N)$ time. On the other hand, the cost of managing the LRU list is independent from the numbre of nodes and it runs in $O(1)$ time.

Table 2. distortion in summary generations

type	3600s	1sx3600	60sx60	1sx60x60	5sx24x30
(gen.)	(1st)	(2nd)	(2nd)	(3rd)	(3rd)
#1 src	0.0	0.0459	0.0441	0.0488	0.0463
#1 dst	0.0	0.0425	0.0312	0.0468	0.0395
#2 src	0.0	0.0085	0.0210	0.0205	0.0213
#2 dst	0.0	0.0115	0.0140	0.0202	0.0204

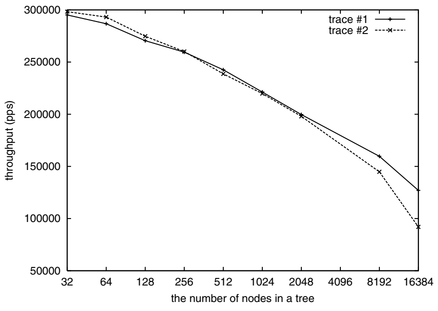


Fig. 10. performance with varying tree size

The impact of the number of nodes to the performance is shown in Figure 10. As the number of nodes in a tree increases, the height of the tree becomes longer and the lookup operation becomes more costly. The execution time is measured to produce both source and destination address profiles with the 3600-second traces on a PentiumIII 700MHz/FreeBSD-4.2. The throughput is shown in packets per second (pps); we simply divide the number of packets in the trace by the user time. Thus, this is not an accurate measure but intended to provide a rough idea about the performance.

The result shows that the profiler can process about 250Kpps with 256 nodes, and about 200Kpps with 2048 nodes. The performance is good enough to monitor a 100Mbps link. In the worst case where a 100Mbps link is filled with 64-byte packets, about 190Kpps is required. As a side note, the forwarding performance of a PC router is much lower; about 80Kpps [18].

5.3 Evaluation Summary

We have evaluated the algorithm using backbone packet traces. The leaf node management using a variant of the LRU replacement policy produces decent summaries. The tree size of 128 or 256 works well even for backbone networks, and its performance is good enough. The algorithm is fairly insensitive to variations in networks, the profiling period length, and summary generations.

The packet traces used for the evaluation are backbone data, and as such, the number of included addresses are considerably larger than enterprise networks. The profiler performs much better in enterprise networks.

6 Application for Traffic Control

Once aggregates are identified and profiled, the profile records can be used for traffic control. There are many possible approaches to control aggregates.

In this paper, we propose an aguri three color marker (aguriTCM), which can be used as a component in a Diffserv traffic conditioner [1]. The aguriTCM combines a profiler with a marker. The profiler part is basically the same and the marker part is intended to be used with the Assured Forwarding (AF) Per Hop Behavior (PHB) [12]. The aguriTCM dynamically identifies and profiles aggregates as already described, and then, marks one of three colors to arriving packets. Here, the colors correspond to DS codepoints assigned for the AF drop precedence levels.

Our use of the Diffserv components is basically local to the node, which differs from the DS domain model of the Diffserv architecture. Our primary target is a protective measure against DoS attacks, and therefore, it makes sense to place a standalone traffic control node at a protection point.

Another major difference is that Diffserv markers are usually configured with traffic profile parameters (e.g., committed target rate) [13], whereas the aguriTCM does not have parameters to specify traffic profiles but automatically adapts to traffic. Again, neither class configuration nor classifier rule is needed for this mechanism.

6.1 aguriTCM

Figure 11 illustrates the traffic control model. Arriving packets are marked by the aguriTCM on the input interface, and preferentially discarded by the AF PHB on the output interface. We use the RIO dropper [10] for the AF PHB.

The aguriTCM degrades the drop precedence level of packets for aggregates whose volume is more than the fairshare. Under long-term congestion, the RIO discards packets according to the drop precedence level assigned to the packet.

One difference in the profiler mechanism is that, at the end of a profiling period, the counters of aggregates are not reset to zero but they are aged. The aging method avoids inaccuracy in the begining of a profiling period and smooths out marking probability.

When the counter value is c at the end of a period, the new value becomes ωc, here ω is the weight for aging. We use 0.5 for ω so that the counters are simply halved. The initial counter value for a period is saved in each node so that the profiler reports the period count by subtracting the initial value.

To find the corresponding aggregate for marking, the aguriTCM first checks whether the entry exists in the previous summary. If the saved initial counter is zero, the entry was not in the previous summary. Then, the aguriTCM goes up

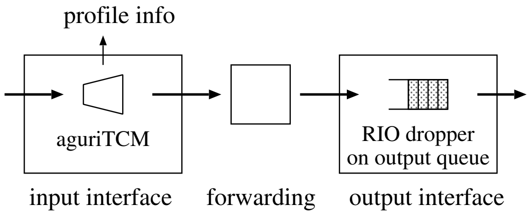


Fig. 11. traffic control model

the tree until it encounters an ancestor with a positive initial counter, and this node is used for marking.

The aguriTCM stochastically demotes the drop precedence of packets if an aggregate uses more than the fairshare. The fairshare is derived from the number of aggregates in the previous summary. When the counter value of an entry is c and the number of aggregates in the previous summary is n, fairshare f is computed as the total count divided by n.

$$f = \frac{\sum c}{n}$$

To demote packets exceeding fairshare f, demotion probability p is computed as

$$p = \begin{cases} \frac{c-f}{c} & \text{if } (c > f) \\ 0 & \text{otherwise} \end{cases}$$

The aguriTCM computes p twice, p_{src} and p_{dst}, independently from the source address and the destination address. An arriving packet is initially considered green. The packet is demoted to red if it is marked by both criteria, and to yellow if it is marked by either criterion. In other words, the packet is marked to red with probability $min(p_{src}, p_{dst})$, and to yellow with probability $|p_{src} - p_{dst}|$.

6.2 Implementation

We have implemented the aguriTCM on the ALTQ framework [5,7] as a Diffserv traffic conditioner component. ALTQ already implements the RIO dropper that supports 3 drop precedence levels. In the current prototype, the aguriTCM always marks packets since packets are dropped only when RIO detects long-term congestion. It could be changed to turn on marking only when the RIO dropper is actively dropping packets to avoid unnecessary marking.

In the prototype, the source and destination address trees are global within the router and shared by multiple instances of aguriTCMs. The profile is produced from all the active aguriTCMs. It allows us to control outgoing packets arriving from different interfaces. Although it is effective only when aggregates share either the incoming interface or the outgoing interface, it covers the majority of the situations requiring the aguriTCM where a router has a single bottleneck or a single fat up-link. If there is less correlation among traffic from different interfaces, it would be better to assign an independent aguriTCM for each interface.

For network monitoring, the aguriTCM writes summaries to a buffer at the end of each profiling period if there is a listener for the aguriTCM device interface. The aguri program in the user space reads the binary summaries through the device interface and produces derivative summaries.

6.3 Preliminary Test Results

The aguriTCM is tested with 7 PCs in a simple configuration shown in Figure 12. 3 senders on the left are on a half-duplex 100baseTX (100Mbps), and 3 receivers

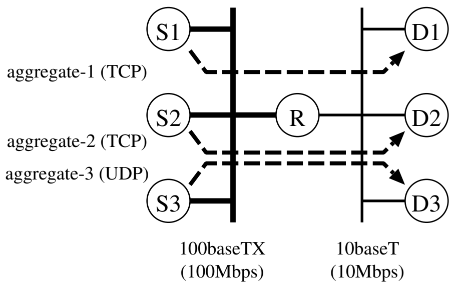


Fig. 12. test configuration with 3 aggregates

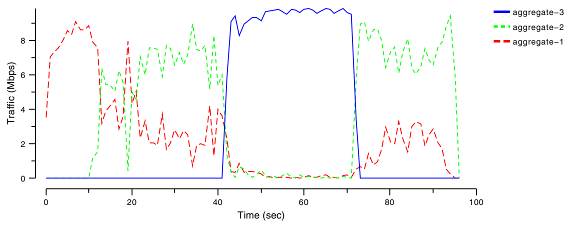


Fig. 13. throughput of aggregates without traffic control

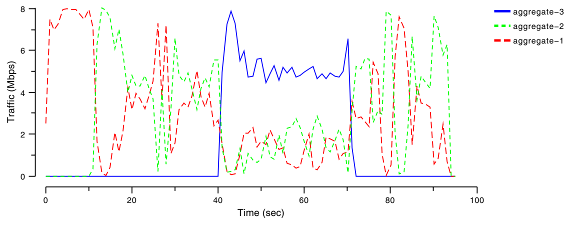


Fig. 14. throughput of aggregates with traffic control

on the right are on a half-duplex 10baseT (10Mbps). The aguriTCM and the RIO dropper are implemented on the router in the middle. The aguriTCM is configured with 1% aggregation threshold, 256 nodes per tree, and 5-second profiling period.

3 aggregates are generated in the tests. Both aggregate-1 and aggregate-2 consist of 4 parallel TCP sessions. Aggregate-3 is a single UDP stream sent at a constant rate of 10Mbps. Aggregate-1 starts at time 0 and aggregate-2 starts at time 10. Aggregate-3 is invoked from time 40 to time 70.

The behaviors of the aggregates are compared with and without traffic control. Figure 13 shows the original behavior and Figure 14 shows the effects of the traffic control. The throughput is measured on the 10baseT link and plotted every second. The plots illustrates (1) resilience in the face of misbehaving flows and (2) fairness among aggregates.

In Figure 13, the UDP forces the TCPs to back off and steals the entire link capacity. On the other hand, in Figure 14, the UDP cannot fill the link after the aguriTCM starts raising the drop precedence of the UDP packets. This result demonstrates the ability to restrict the bandwidth use of misbehaving flows.

In Figure 13, the bandwidth share of the 2 TCP aggregates is not fair even when aggregate-3 is not active. It is improved in Figure 14 since packets are dropped from the aggregates using more than the fairshare. Although it is observed that the TCP throughputs go up and down in the plot, it would be improved if there are more aggregates or TCP implements better recovery mechanisms such as NewReno and SACK.

Figure 13 also shows the problem of unfairness among TCP sessions. It is common that competing TCPs have unequal bandwidth share due to the differences in various factors such as RTT, TCP implementation, and CPU power or other hardware. Among other things, unfairness by RTT is inherent in the TCP mechanism because a session with smaller RTT opens up the congestion window more quickly. The aguriTCM improves this situation since flows in a prefix-based aggregate are likely to have similar RTT.

This particular case in our test is caused by the different network cards used at $D1$ and $D2$. The network card of $D1$ seems to implement a more conservative collision recovery than $D2$. As a result, the TCPs in aggregate-1 experience ACK compression on the reverse path and frequently stall for short periods. If we use the same network cards for $D1$ and $D2$, their share becomes equal.

7 Conclusion

We were in need of an adaptive traffic profiler to track long-term trend and to discover problems in our backbone network, and have developed a tool called aguri. Aguri adapts itself to spatial traffic distribution by aggregating small volume flows into aggregates, and achieves temporal aggregation by creating a summary of summaries applying the same algorithm to its outputs. We have been monitoring our network using aguri since February 2001, and found it useful for network operation.

We have also presented a technique to combine an aggregation-based traffic profiler with a preferential packet dropping mechanism in order to protect the network from DDoS attacks and to provide rough fairness among aggregates. The preliminary test results on our prototype look promising but further investigation and parameter tuning are needed.

The implementation of aguri along with the related tools and other information is available from `http://www.csl.sony.co.jp/~kjc/software.html`.

References

1. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, Internet Engineering Task Force, December 1998.

2. N. Brownlee. Traffic flow measurement: Experiences with NeTraMet. Request for Comments 2123, Internet Engineering Task Force, March 1997.
3. N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: Architecture. Request for Comments 2722, Internet Engineering Task Force, October 1999.
4. Kenjiro Cho. Tele Traffic Tapper. `http://www.csl.sony.co.jp/~kjc/software.html`, 1996.
5. Kenjiro Cho. A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers. In *USENIX 1998 Annual Technical Conference*, pages 247–258, June 1998.
6. Kenjiro Cho. Flow-valve: Embedding a safety-valve in red. In *Global Internet Symposium, Globecom*, pages 1753–1762, December 1999.
7. Kenjiro Cho. *The Design and Implementation of the ALTQ Traffic Management System*. PhD thesis, Keio University, January 2001.
8. Kenjiro Cho, Koshiro Mitsuya, and Akira Kato. Traffic data repository at the WIDE project. In *USENIX 2000 Annual Technical Conference: FREENIX Track*, pages 263–270, June 2000.
9. Kimberly C. Claffy, Hans-Werner Braun, and George C. Polyzos. A parameterizable methodology for internet traffic flow profiling. *IEEE Journal of Selected Areas in Communications*, 13(8):1481–1494, 1995.
10. D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4), August 1998.
11. Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transaction on Networking*, 7(4):458–472, August 1999.
12. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597, Internet Engineering Task Force, June 1999.
13. J. Heinanen and R. Guerin. A two rate three color marker. RFC 2698, Internet Engineering Task Force, September 1999.
14. V. Jacobson, C. Leres, and S. McCanne. tcpdump. `ftp://ftp.ee.lbl.gov/`, 1989.
15. V. Jacobson, C. Leres, and S. McCanne. libpcap. `ftp://ftp.ee.lbl.gov/`, 1994.
16. Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and K. Claffy. The architecture of the CoralReef internet traffic monitoring software suite. In *PAM 2001*, Amsterdam, The Netherlands, April 2001.
17. Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *draft paper*, February 2001.
18. Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. The Click moduler router. In *Proceedings of SOSP'99*, pages 217–231, Kiawah Island Resort, SC, December 1999.
19. Tobias Oetiker. RRDtool: Round Robin Database Tool. `http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/`.
20. Tobias Oetiker. MRTG: The multi router traffic grapher. In *USENIX LISA Conference*, pages 141–147, Boston, MA, December 1998.
21. Dave Ponka. FlowScan: A network traffic flow reporting and visualization tool. In *USENIX LISA Conference*, New Orleans, LA, December 2000.
22. John T. Robinson and Murthy V. Devarakonda. Data cache management using frequency-based replacement. In *SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 134–142, May 1990.
23. Keith Sklower. A tree-based packet routing table for berkeley UNIX. In *USENIX Winter Conference*, Dallas, Texas, January 1991.
24. S. Waldbusser. Remote network monitoring management information base. Request for Comments 1757, Internet Engineering Task Force, February 1995.

Traffic Handling in AQUILA QoS IP Network

A. Bak[1], W. Burakowski[1], F. Ricciato[2], S. Salsano[2], and H. Tarasiuk[1]

[1] Institute of Telecommunications, Warsaw University of Technology,
ul. Nowowiejska 15/19, 00-665 Warsaw (Poland)
{bak, wojtek, halina}@tele.pw.edu.pl
[2] CoRiTeL – v. Anagnina 203, 00040 Roma (Italy)
{ricciato, salsano}@coritel.it

Abstract. The paper describes the traffic handling mechanisms implemented in the AQUILA pilot QoS IP network [10]. The AQUILA project enhances the DiffServ architecture concept [1,2,3] by adding new functionality for admission control and resource management as well as by defining new set of network services. Each network service is optimised for specific type of traffic (e.g. non-reactive and reactive) and has its own traffic handling mechanisms. Exemplary measurement results verifying the effectiveness of AQUILA approach for providing QoS are also included.

1 Introduction

Offering QoS (Quality of Service) in IP networks is of strategic importance for Internet Service Providers. The AQUILA[1] project tackles this challenge, aiming at the definition of a QoS architecture for IP. The problem involves the following aspects: (1) network architecture and traffic control (e.g. definition of QoS signaling, interaction with routing protocols, etc.) (2) traffic handling (e.g. packet scheduling, admission control algorithms, etc.), and (3) management aspects (e.g. user subscription to QoS services, accounting, billing, etc.). AQUILA has focused on a subset of these topics, mainly covering architectural and traffic handling aspects, while, for example, the management and billing issues were not covered.

This paper mainly deals with the traffic handling issues in AQUILA network. A detailed description of the overall AQUILA architecture one can find in [6]. Traffic handling for QoS is rather complex issue involving a set of mechanism operating on different time scales, from milliseconds (packet scheduling) to hours and days (resource provisioning, traffic engineering). AQUILA organizes all the traffic handling aspects in a single vision. Currently, after 15 months from the start of the project, the AQUILA project is running its first trial and it has started the specification for the second project phase. The testbed that has been developed within the project provides a great opportunity to see a running complete QoS architecture [10]. The goal of the paper is to present the specification of the traffic handling mechanisms and

[1] AQUILA (Adaptive Resource Control for QoS using an IP-based Layered Architecture) IST-1999-10077 is an European research project partially funded be the IST programme

to start analysing the project results and the experiences that are being maturated in the testbed.

Section 2 introduces us to the AQUILA architecture and investigated concepts, with a special focus on the relationships between the traffic handling mechanisms reffering to different time scales. Sections 3 an 4 describe the traffic control mechanisms operating at the packet level and at the flow/aggregate level, respectively. Section 5 presents first experimental results, mainly covering the traffic control at the packet level. Finally, section 6 summarises the paper.

2 Overview of Traffic Handling Approach

This section provides an overview of the AQUILA architecture and concepts. Let us briefly recall the rationale and the architecture of AQUILA. Two important aspects of QoS are QoS *guarantees*, and QoS *differentiation*. In order to provide QoS differentiation, a limited set of Network Services (NS) have been defined in AQUILA project, which represent the services sold by the provider to its customers: Premium Constant Bit Rate (PCBR), Premium Variable Bit Rate (PVBR), Premium Multimedia (PMM), Premium Mission Critical (PMC) and Standard Best Effort (STD). Each NS is meant to support a class of applications with substantially similar *requirements* and *traffic characteristics*. The Network Services are internally mapped by the operator into a set of Traffic Classes. The Traffic Classes use Diffserv based packet handling mechanisms. For providing QoS guarantees the ISP must somehow regulate the volume of traffic submitted to the network, regarded as a limited set of resources. In the AQUILA approach this is accomplished by a distributed layer: the Resource Control Layer (RCL). The RCL components are the End-user Application Toolkit (EAT), located at the end-user site, the Admission Control Agent (ACA), located at the edge of the network (e.g. in the Edge Routers), and the Resource Control Agent (RCA), which is logically a centralized entity within the network itself. The RCL embeds different mechanisms to regulate the traffic at different time-scales – Initial Provisioning, Dynamic Resource Pool and Admission Control. Note that in the first phase of the AQUILA project, the focus is on QoS in a single domain.

Let us now turn our attention on the *traffic handling* mechanisms in AQUILA. Traffic handling is used here as a general term for a set of coordinated mechanisms operating at different time scales:

- **Traffic control** refers to the mechanisms operating at milliseconds time scale like packet scheduling, policing, queue management.
- **Admission control** refers to the algorithms to decide about the acceptance of a new flow in the network, operating at the time scale of seconds to tens of minutes.
- **Resource Pools** refers to the algorithm for short term resource redistribution, to cope with local fluctuations in offered traffic, operating at the time scale of tens of minutes to hours.
- **Provisioning** refers to the algorithm for medium/long term resource allocation and redistribution, operating at the time scale of hours to days.

The relationships among these logical components (Provisioning, Resource Pools, Admission Control and Traffic Control) are briefly described in the following. A very high-level view of the process that enables QoS in the AQUILA architecture is shown in Fig.1, while Fig. 2 presents a simplified pictorial view of the relationships between the different mechanisms. More details are provided in the next sections.

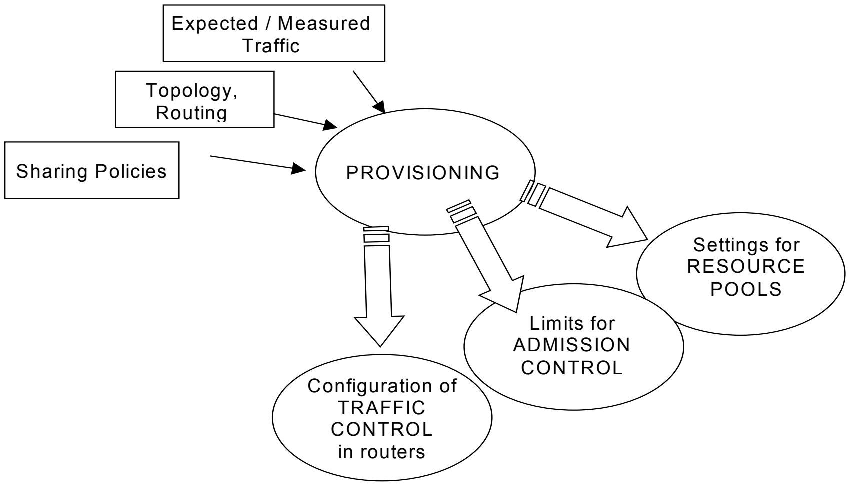

Expected / Measured Traffic
Topology, Routing
Sharing Policies
PROVISIONING
Settings for RESOURCE POOLS
Limits for ADMISSION CONTROL
Configuration of TRAFFIC CONTROL in routers

Fig. 1. Enabling QoS in the AQUILA architecture

The Provisioning phase is run off-line before the network operation, and gives the required input to the RCL elements as well as configuration values for setting the router parameters. The initial provisioning algorithm takes as an input global information about the topology, the routing (costs of links), the expected traffic distribution between Edge Routers for each Traffic Class, and any further constraints on the link bandwidth sharing between TCLs. It performs a sort of global computation and produces as output:

- the expected amount of traffic for each Traffic Class on each link, called *provisioned rate*. This is used for the router configuration, i.e. to chose the appropriate setting for the scheduling / queuing parameters (WFQ weights, WRED thresholds) at each router interface.
- the Admission Control Limits for each Traffic Class at each Edge Router. This is used by AC algorithms during the operation phase..
- Definition of the Resource Pools sets (RP will be discussed in section 0).

The Traffic Control mechanisms define how the packets of the different classes are handled by the Edge and Core Routers in the AQUILA network. They includes traffic conditioning (also referred to as *policing*), that is enforced at ingress ERs only, and scheduling / queuing algorithms, implemented at any router interface.

The configuration of the scheduling / queuing mechanism is “static”, i.e. the relevant parameters are configured in the routers at start up. An off-line procedure computes these parameters starting from the provisioned rates produced by the Initial Provisioning algorithm.

Obviously, the configuration of per-flow traffic conditioning parameters at the ingress edge is done run-time according to the admitted requests. Details on the traffic conditioning mechanisms will be given in section 0.

The Admission Control procedure is intended to restrict traffic in order to avoid congestion. The AC procedure is operated on-line, but the AC reference limits, or AC Limits, are computed during the off-line initial provisioning phase and configured during the start-up phase.

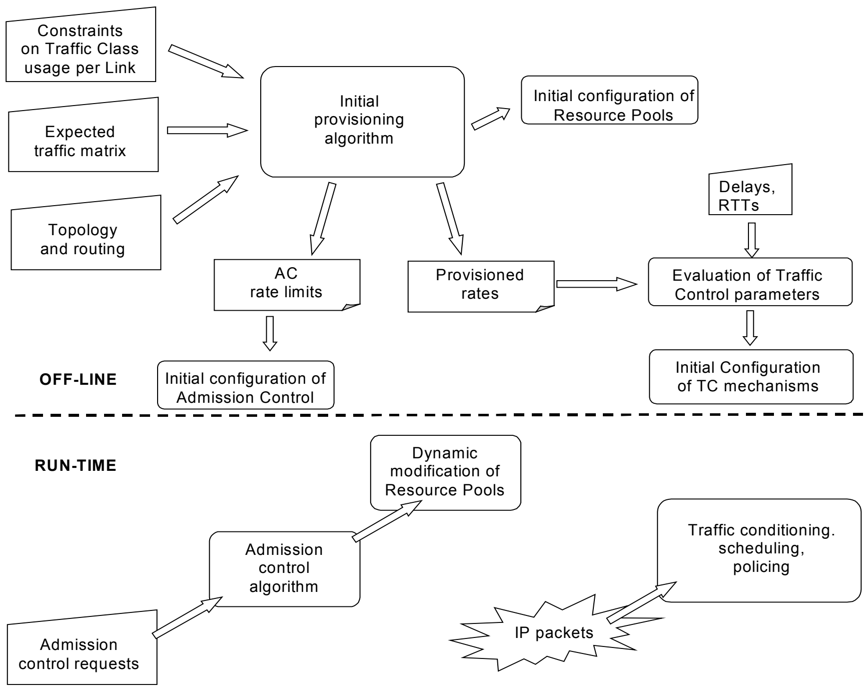

Constraints on Traffic Class usage per Link
Expected traffic matrix
Topology and routing
Initial provisioning algorithm
Initial configuration of Resource Pools
Delays, RTTs
AC rate limits
Provisioned rates
Evaluation of Traffic Control parameters
OFF-LINE
Initial configuration of Admission Control
Initial Configuration of TC mechanisms
RUN-TIME
Dynamic modification of Resource Pools
Traffic conditioning. scheduling, policing
Admission control algorithm
IP packets
Admission control requests

Fig. 2. Initial Provisioning, Traffic Control and Admission Control

The assignment of AC Limits to each Edge Router for each TCL represents a resource assignments to the relevant traffic aggregates. As the AC Limits are computed based on the expected offered traffic at each ER, some deviation can occur during the operation phase between the actual offered traffic and the resource distribution between ERs. The Resource Pools mechanism represent a way to dynamically change the AC Limit to some extent, so as to dynamically track short term fluctuations in traffic requests. Such mechanisms are based on the concept of RP, which are sets of ER that can exchange resources with each other. Such sets are defined during the Initial Provisioning phase.

3 Traffic Handling Mechanisms at Packet Level: The AQUILA Traffic Classes

From the network point of view, the differentiation at the service level into Network Services (NS) naturally introduces a differentiation at the packet handling level. In particular, the implementation of relative priorities between the packets, both for the access to the transmission channel and/or to the buffer space, is exploited to differentiate the delivered end-to-end delays and loss probability. Although the *prioritisation* induced by the applications requirements is a key component of the traffic handling at the packet level, it is not the only reason to introduce packet handling differentiation inside the network: the other factor is the need for *separation* between traffic flows with dramatically different characteristics. As an example, it is well recognized that closed loop flows (typically TCP and in general TCP-friendly flows) should not compete on the same resources with open loop flows (typically UDP). Separation between the two can be achieved e.g. by using different queues served with some bandwidth sharing mechanism (WFQ scheduling).

Prioritisation and separation represent the two main aspects that any design of packet handling mechanisms must take care of. Beyond that, one should also take into account the excess packets treatment and, eventually, those advanced queue management schemes meant to enhance the performances of reactive traffic (typically RED for TCP).

In designing the packet handling mechanisms inside the router, AQUILA has taken into account all of that: AQUILA has defined a set of 5 Traffic Classes (TCL). At each Edge Router (ER), each TCL is assigned a portion of “resource”, i.e. a bandwidth value, which is meant to limit the maximum amount of traffic that the ER can inject into the network for the specific TCL. As this value will be used by the Admission Control algorithm to decide about the acceptance of new flows to the relevant TCL, it will be denoted by AC Limit. The proposed AC algorithms for each TCL will be discussed in section 3.1. Each TCL is associated a different queue in the router output interface, and a bandwidth portion on each link. All the queues except the first one are served by a WFQ scheduler, thus each TCL is associated to a WFQ weight. The queue dedicated to TCL-1 is served with strict priority over the others. Fig. 3 shows the inter-TCL scheduling scheme. It should be considered that such inter-TCL scheduling scheme applies to high speed router interfaces. In fact for low-speed interfaces (few hundreds kilobits per second), which can be found in the access network section, it is not reasonable to statically partition the (little) available bandwidth between the TCLs. To cope with this problem, AQUILA TCLs are implemented in a different way on low-speed interfaces. However for sake of space we will not deal with low-speed interface in this paper. We refer to [7] for further details on this topic.

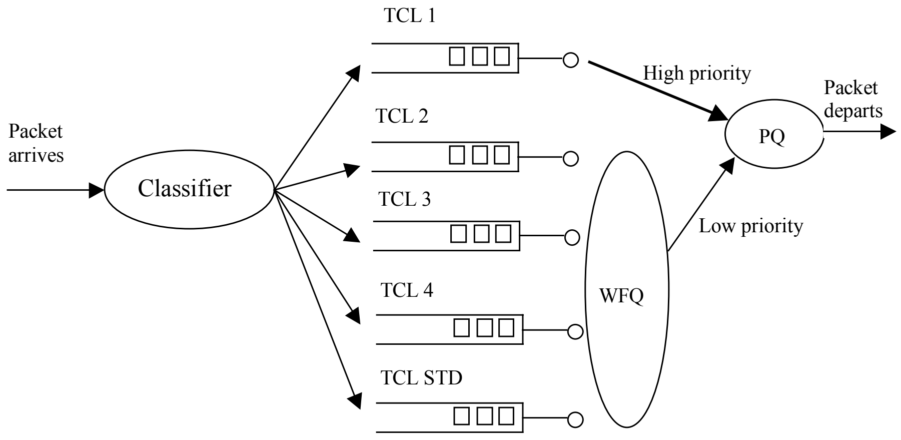


Fig. 3. Design of router output port

TCL-5 is intended to support the Standard Service (STD), i.e. the traditional best-effort traffic. The traffic accessing the STD service is not delivered any QoS and is not regulated by any AC and/or policing function inside the ER. Nevertheless, a non-null amount of bandwidth will be guaranteed to this traffic on each link, accordingly the WFQ weight for TCL-5 will be non-zero.

TCL-1 and TCL-2 are intended to support non-reactive (open loop) traffic with stringent QoS requirements. In particular TCL-1 will be characterized by very high QoS performance (very low delay and very low losses), accomplished by a conservative AC scheme: no statistical multiplexing is achieved within this TCL, and the AC algorithm will exclusively base on the flows peak-rate which is declared by the applications. In the AQUILA architecture TCL-1, which is somehow similar to the EF PHB defined in [2] will exclusively support the PCBR service. Typically, TCL-1 will be entered by flows with small to medium packet size (< 256 B) and not too large peak-rate, as typically originated by real-time streaming applications like VoIP, etc. On the other hand, TCL-2 will deliver a lower QoS level (low delay and low losses) to those streaming application with high emission rate variability and/or large packets: the AC for TCL-2 scheme allows for some degree of statistical multiplexing, thus the mean rate of the flows must be taken into account in the AC algorithm along with the peak-rate. TCL-2 will mainly support the PVBR network service.

TCL-3 and TCL-4 are dedicated to reactive flows (TCP and TCP-like). In particular, TCL-3 will support PMM service and will take long-lived TCP connections (for long file transfers) or other adaptive application flows (audio/video download, adaptive video). These flows are typically greedy, as they continue to expand the emission rate until congestion is reached. TCL-4 instead will support PMC service and will receive non-greedy elastic flows, typically short-lived TCP connections originated by some critical transaction application (e.g. finance) or interactive games. Note that separating long-lived and short-lived TCP connections into different classes, thus avoiding direct competition on the same resources (the WFQ scheduler acts as a sort of arbiter for the bandwidth) prevents the former from starving the latter. Further details about queuing mechanisms as well as traffic conditioning for each TCL are given below.

It can be noted that there is a one to one mapping between the defined TCLs and the NSs. This should be considered as a sort of coincidence, as there is no one-to-one mapping requirement between NSs and TCLs but the provider has a degree of freedom in this mapping. Nevertheless, for sake of simplicity this degree of freedom has not been exploited in the current specification. It is expected that as the AQUILA approach will evolve, new NSs could be defined while keeping the same set of supported TCLs. On the other hand, there is also the possibility that, based on the ongoing trial experiences, a reduction in the number of TCL could be envisioned: in particular the need of two different TCLs for supporting streaming traffic is under investigation, therefore TCL-1 and TCL-2 could be merged into one single TCL.

Note that the AC function in the network is associated to the TCLs, rather than to the NS. In fact the resources within the network (corresponding to the AC Limits inside the ERs), are assigned on a per-TCL basis.

As a final remark, we are of course aware that a lot of effort is ongoing within the IETF to standardize Per Hop Behaviors (PHB). The scope of AQUILA is not to define new PHBs that will replace the ones defined in the IETF. The main focus of AQUILA is to study the whole QoS picture, from millisecond scale (Traffic Classes) up to long term scale (Provisioning). It was not possible at the AQUILA design time to rely on a stable definition of Diffserv PHBs and it was even farther the possibility to have Diffserv compliant implementation in routers. Therefore it was chosen to design the set of Traffic Classes based on Traffic control mechanism available in the routers. It is possible to re-map the AQUILA TCLs onto the “standard” Diffserv PHBs (EF, AF), but this is out of the scope of this work.

3.1 Traffic Control at Packet Level

This subsection provides further details about the packet scale traffic control mechanisms, i.e. traffic conditioning and queue management, for each TCL.

TCL-1 and TCL-2 are both dedicated to non-reactive (UDP) flows with stringent QoS requirements. According to that, a “severe” purely dropping traffic conditioning at ingress point is associated to both, i.e. all packets exceeding the declared profile are discarded. The traffic profile for TCL-1 is described in terms of a Single Token Bucket, limiting the flow peak rate. The traffic profile for TCL-2 is described in terms of a Dual Token Bucket, controlling both the peak and mean rate of the flow. This is consistent with the definition of AC algorithms: AC for TCL-1 is based only on peak rate, while AC for TCL-2 takes into account the mean rate also to achieve better multiplexing gain (see section 0 for details). For both TCL-1 and TCL-2, queues at router interfaces are simply of FIFO drop-tail type.

TCL-3 is dedicated to long-lived TCP controlled flows. A Single Token Bucket descriptor is used to declare the mean rate only. Traffic conditioning at ingress point is based on 2 colours marking: out-of-profile packets are not discarded but simply marked as such with a different DSCP value. At router interfaces, the TCL-3 queues employs a WRED management algorithm with two different sets of parameters (minth, maxth, maxp) for in-profile and out-of-profile packets.

TCL-4 is dedicated to short-lived non-greedy TCP-controlled flows with low bandwidth requirements. Dual Token Bucket descriptor is used to declare mean and peak rate. Traffic conditioning and queue management are similar to TCL-3.

Finally, simple FIFO drop-tail queues are used for the STD TCL.

4 Traffic Handling Mechanisms at Flow and Aggregate Level

In the previous sections we have introduced the concept of NS and TCL, which aim at achieving QoS *differentiation* respectively at the service and at the traffic level. This chapter is now focused on the mechanisms that AQUILA implements to deliver QoS *guarantees*. For QoS to be delivered, the amount of traffic entering the network must be somehow regulated. To this purpose, AQUILA considers a combination of three mechanisms at aggregate level: Initial Provisioning, Dynamic Resource Pool and Admission Control. The following of this section discusses these mechanisms in a top-down fashion. These mechanisms are complementary to those discussed above operating at the packet level (e.g. traffic conditioning).

4.1 The Initial Provisioning Phase

For each TCL j each ER i is assigned a certain amount of bandwidth $l_i^{(j)}$, called AC Limit, which is used by the admission algorithm in the ACA responsible for ER i as a reference limit to the traffic it can accept for TCL j. The value of the AC Limit as well as the bandwidth demands for each active reservation are logically stored in the ACA. The initial computation of the set $\{l_i^{(j)}\}$ for each TCL and ER, called the Initial Provisioning phase in AQUILA, can be done off-line and must take into account the following input information:

a. The complete network topology, included the link capacities and link cost (as used by IGP routing for computation of shortest path).
b. The long- term expected traffic matrices, i.e. the expected spatial distribution of traffic for each TCL between source/destination router pair.
c. Some bandwidth sharing policies between TCL, e.g. “no more than 10% of link capacity allocated to TCL-1”, or “no less than 30% of link capacity allocated to best effort traffic (TCL-5)” and similar.

In the simplest scenario classical IGP routing is used within the network (e.g. OSPF, RIP, etc.), and the route for each source/destination pair is unique and constrained to the shortest path. Nevertheless in more advanced scenarios, particularly when MPLS is used for Traffic Engineering purposes, the routing itself is not strictly constrained to shortest paths, and the paths computation could be made jointly with the AC Limits computation. In this case there would be room for some sort of *global optimisation* during the Initial Provisioning phase, whose output would be the set of paths along with the set of AC Limits. This additional perspective has been left out of the scope of this paper. Also note that fault recovery aspects are not covered by AQUILA.

4.2 The Resource Pool

The scheme described up to now meets the requirement to process each RR locally at the ACA, but at the cost of a high rigidity in the resources distribution. In fact per ER ("horizontal") as well as per TCL ("vertical") bandwidth assignment is done statically in the initial provisioning phase. This scheme is not able to dynamically track fluctuations and deviations of the actual offered traffic from the expected one. In order to gain a more dynamical behaviour, it would be desirable to have dynamical sharing of resources between ER ("horizontal" sharing) and/or between TCL ("vertical" sharing) to some extent. AQUILA introduces the concept of "Resource Pool" (RP) to allow for some degree of dynamical sharing of resources between ER ("horizontal" sharing) only. Further introduction of some form of resource sharing between TCLs ("vertical" sharing) is envisioned as a further extension to the model.

In order to explain the RP concept, consider the case that an amount of Resource Reservations for TCL j are being rejected at some ER x due to consumption of the relevant AC Limit, while at the same time some other ER y does not fulfill its resource budget due to lack of demand. In this case it would be desirable to dynamically shift a portion of the resource budget from y to x, by increasing $l_x^{(j)}$ and decreasing $l_y^{(j)}$. In this example we will denote ER y as the "donor". The concept of RP arises from the general consideration that not any ER can be a meaningful "donor" for any other ER: there are constraints between AC Limits due to the topology and the traffic distribution, mainly related to the presence of *bottlenecks*. Thus, a RP identifies a set of ERs that can be donors to each other: inside a RP the ERs can exchange between each other the assigned bandwidth for a given TCL. Application of the RP concept is straightforward in the case a set of ERs are connected in a star to a core router. The case is depicted in Fig. 4, which will be used to illustrate the dynamical resource distribution algorithm inside the RP.

The above approach can be hierarchically extended to build RP whose elements are not ERs but other RPs. The case is depicted in Fig. 4, where RP "C" is composed by node 9 as root and RPs "A" and "B" as leaves. The hierarchical RP approach can be straightforwardly applied in those networks whose external areas present a hierarchically structured topology, which is expected to be a quite common case in practice.

4.3 Admission Control

In order to control the access to the network, the user applications must "ask for permission" before sending traffic to the network. This permission is granted by a Reservation Request (RR) sent to the network. The task of sending RRs is covered by the EAT module at the user side, which represent a sort of interface between the (legacy) applications and the QoS network. The RR sent from the EAT is processed at the network side by the ACA, which can accept or reject the flow depending on the *available resources*, i.e. on the profiles of the already admitted flows (for the same TCL, at the same ER) and on the relevant AC Limit.

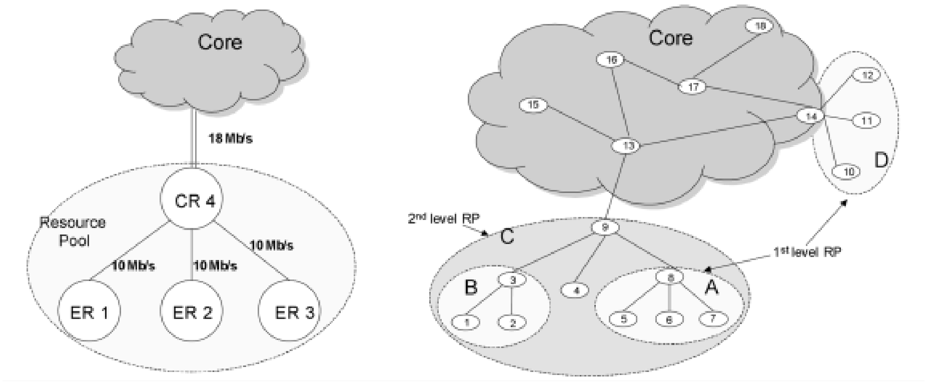


Fig. 4. Resource Pool

The proposed AC algorithms are derived from the results developed in the context of ATM traffic control. The request for network resources is accepted/rejected based on the traffic descriptors provided by the user. In the AQUILA architecture the admission decision is made only at the network ingress and, in some cases, at the egress point. This makes the AC decision more critical, as link-by-link verification of resource availability is not possible. To perform the admission control at the ingress or egress the single link model was considered with capacity C (AC limit) and buffer size B. Furthermore, for simplicity, the isolation between all traffic classes was assumed. In fact, the TCL-1 class has impact on other classes as it is served with the highest priority (see section 3). Whenever below are mentioned parameters C or B, they correspond to the capacity and buffer size dedicated to serve the given traffic class. In the following, details about the AC used in Aquila are provided.

Admission control algorithm for TCL-1

The TCL-1 class uses peak rate allocation scheme [4]. A flow in this class is characterised by the parameters of single token bucket algorithm that correspond to the peak bit rate (PBR) and peak bit rate tolerance (PBRT). In the AQUILA architecture, the TCL-1 traffic is served with the highest priority. Taking this into account, it can be assumed that the TCL-1 streams have negligible packet delay variation [11]. Consequently, the worst case traffic pattern for the superposition of a number of TCL-1 flows takes the form of poissonian stream (with the mean rate equal to the sum of the PBR parameters of the particular flows).Let us assume that the capacity dedicated for TCL-1 class is C_1. In the case, when N1 flows with $\{PBR_1, PBR_2, \ldots, PBR_{N1}\}$ are currently in progress, a new flow declaring PBR_{new} as its peak rate is admitted if the following condition is satisfied:

$$PBR_{new} + \sum_{i=1}^{N1} PBR_i \leq \rho C_1 \qquad \textbf{(1)}$$

Parameter ρ (ρ<1) specifies the admissible load of capacity allocated to the TCL-1 class. The value of ρ is calculated from the analysis of M/D/1/B system taking into account the assumed target packet loss ratio and buffer size [5].

Admission control algorithm for TCL-2

In case of TCL-2 traffic class the REM (*Rate Envelope Multiplexing*) multiplexing scheme is assumed for guaranteeing low packet delay [4]. Therefore, the only QoS parameter that requires concern is the packet loss rate. In the REM multiplexing the buffer (relatively small) has to be dimensioned for absorbing, so called, the packet scale congestion (simultaneous arrival of packets from different sources). For this purpose the N*D/D/1 queuing system analysis is useful. In the TCL-2 class, each flow is characterised by the parameters of the Dual Token Bucket: the peak bit rate (PBR) jointly with the peak bit rate tolerance (PBRT) and the sustainable bit rate (SBR) jointly with the burst size (BSS). It is commonly believed that the worst-case traffic pattern for given values of PBR, SBR and BSS is of ON/OFF type. The proposed admission method for TCL-2 is based on the notion of effective bandwidth. One may find a number of methods for calculating effective bandwidth [4]. For its simplicity, the methods proposed in [12] was chosen for AQUILA. In this method the value of effective bandwidth, Eff(.), is calculated on the bases of PBR, SBR and BSS parameters taking into account the target packet loss rate.

Let us assume that the capacity dedicated for TCL-2 class is C_2. In the case, when N2 flows with {Eff(1), Eff(2), …, Eff(N2)} are currently in progress, a new flow with Eff(new) is admitted if the following condition is satisfied:

$$Eff(new) + \sum_{i=1}^{N2} Eff(i) \leq C_2 \qquad \textbf{(2)}$$

Admission control algorithm for TCL-3

In the case of TCL-3, each flow is characterised by parameters of single token bucket algorithm that correspond to the sustained bit rate (SBR) and the burst size (BSS). The BSS values for typical flows accessing TCL-3 are expected rather large, allowing for a high variability of submitted traffic. As a difference with other TCLs, the declaration of the PBR parameter is omitted, so that the PBR will be only limited by the access links speed and intrinsic TCP dynamics.

Taking into account that the traffic flows submitted to TCL-3 class are TCP-controlled, only rough QoS guarantees are assumed to be provided. Therefore, an admission control method that maximises the network utilisation can be employed in this case.

Let us assume that the capacity dedicated for TCL-3 class is C_3. In the case, when N3 flows with {(SBR_1, BSS_1), (SBR_2, BSS_2), …, (SBR_{N3}, BSS_{N3})} are currently in progress, a new flow declaring ($\mathrm{SBR_{new}}$, $\mathrm{BSS_{new}}$) is admitted if the following condition is satisfied:

$$SBR_{new} + \sum_{i=1}^{N3} SBR_i \le C_3 \tag{3}$$

Admission control algorithm for TCL-4

In the TCL-4, a flow is characterised by parameters of dual token bucket algorithm, similarly as in the case of TCL-2 class. The proposed admission control algorithm aims to provide the service rate that will guarantee virtually no packet losses.

Let us assume that the capacity dedicated for TCL-4 class is C_4. In the case, when N4 flows with {Eff(1), Eff(2), …, Eff(N4)} are currently in progress, a new flow with Eff(new) is admitted if the following condition is satisfied:

$$Eff(new) + \sum_{i=1}^{N4} Eff(i) \le C_4 \tag{4}$$

The effective bandwidth in this case is calculated by (see [13]):

$$Eff(.) = \max\left\{ SBR, \frac{PBR \cdot T}{B/C + T} \right\} \tag{5}$$

where

$$T = \frac{BSS}{PBR - SBR} \tag{6}$$

5 Experimental Results

This section presents experimental results [10] illustrating quality of service offered by two selected TCLs defined in AQUILA i.e. TCL-1 and TCL-3. These classes are designated for different traffic types, for streaming (PCBR NS) and elastic (PMM NS) traffic.

The experiments were carried out in the AQUILA testbed (see Fig.5), installed in Polish Telecom at Warsaw. The test topology consists of 8 CISCO routers (of different types) connected in the form of the chain for achieving large number of hops. The end terminals are connected to the edge routers by Ethernet ports. The access links (deployed between the edge router and the first core router) are of rather low speed (2 Mbps). The higher capacity links (10 and 155 Mbps) connect the core routers. The following types of routers are installed in the testbed: two edge routers - 1605 (aq1605_2) and 3640 (aq3640_4), 6 core routers - 3640 (aq3640_1, aq3640_2, aq3640_3,) and 7507 (aq7507_1, aq7507_2, aq7507_3). Details about router configuration values can be found in [10].

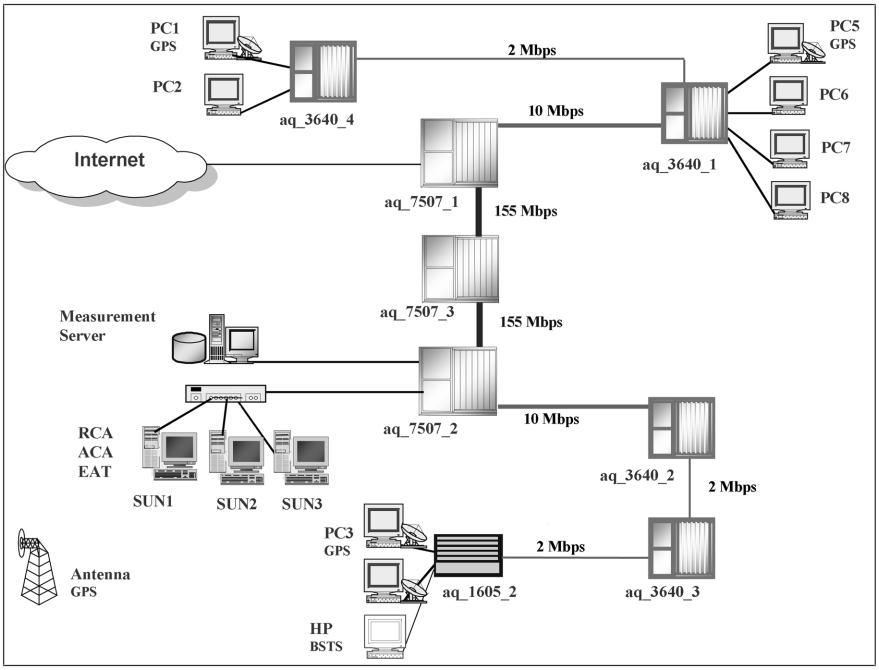


Fig. 5. AQUILA testbed configuration. PC1-8 – terminals, SUN1-3 – SUN Work Stations with implemented RCA, ACA and EAT modules, aq_1605, aq_3640, aq_7507 – CISCO routers

5.1 Results for TCL-1

TCL-1 class was tested assuming that the packet traffic submitted to this class is the maximum traffic allowed by the admission control. The test traffic (TCL-1) was modelled by a Poisson process with constant packet length. Such traffic represents the worst case of the superposition of large number of CBR streams. To take into account the impact of other class traffic on the TCL-1 packets, background traffic was added. The background traffic was sufficient to load the rest of link capacity not dedicated to TCL-1.

In the following experiments, 200 kbps of the access link capacity was reserved for TCL-1. Furthermore, the TCL-1 buffers in the routers were set to 5 packets to guarantee low packet delay requirements. The performance of TCL-1 was validated assuming target packet loss ratio (Ploss) to be 10^{-2}. According to the specified admission control algorithm the maximum admissible load in this case (acc. to the M/D/1 system analysis [4]) is ρ=0.685, what is equivalent to 137 kbps.

The foreground traffic was transmitted between PC1 and PC3 terminals (see Fig. 5) while the background traffic was generated only on the access link (between aq_1605_2 and aq_3640_3 routers – see Fig.5). The background traffic was created by a mix of packets with different lengths: 7% with 44 bytes, 21% with 256 bytes and

72% with 1280 bytes. Both foreground and background traffic was transmitted using UDP protocol.

The characteristic of packet loss rate as a function of the TCL-1 traffic volume is reported in Fig.6. One can observe that the measured value of packet loss rate are in the range of 10^{-5} and they are significantly below the assumed target value 10^{-2} even for the load above the admission region (133 kbps). This is rather expected result since the TCL-1 traffic is served with higher priority with the effective service rate of 2 Mbps (instead of 200 kbps as was assumed for the admission control algorithm). Anyway, increasing TCL-1 traffic above the assumed limit (133 kbps) is a bit dangerous since this can degrade the quality experienced by packets carried in low priority classes (e.g. TCL 2). The recommended admissible load of TCL-1 traffic is approximately 10 % of total link capacity.

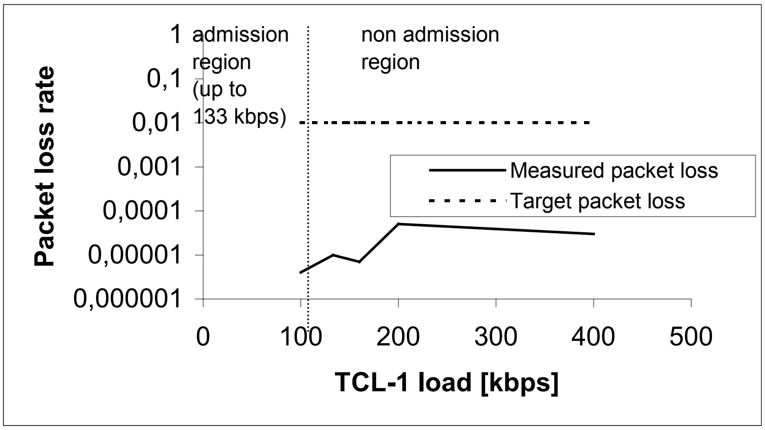


Fig. 6. Packet loss rate vs. TCL-1 traffic load

The characteristics of one-way packet delay as a function of TCL-1 packet length are depicted of Fig.7. These curves were measured assuming that Poisson traffic with the rate equal to 133 kbps (up to admission limit) was submitted to TCL-1. In this case, the background traffic was of ON/OFF type and submitted independently to each intermediate link (see Fig.5), with the peak rates equal to the appropriate link rates. Such type of traffic produces maximum packet delay for the foreground traffic and this is caused by transmission buffer implemented in CISCO routers [10]. The maximum observed delay is below 50 ms. This value is acceptable for applications like for example voice transmission, that tolerates delay in the order of 150 ms, considering that codec and packetisation delay has to be added.

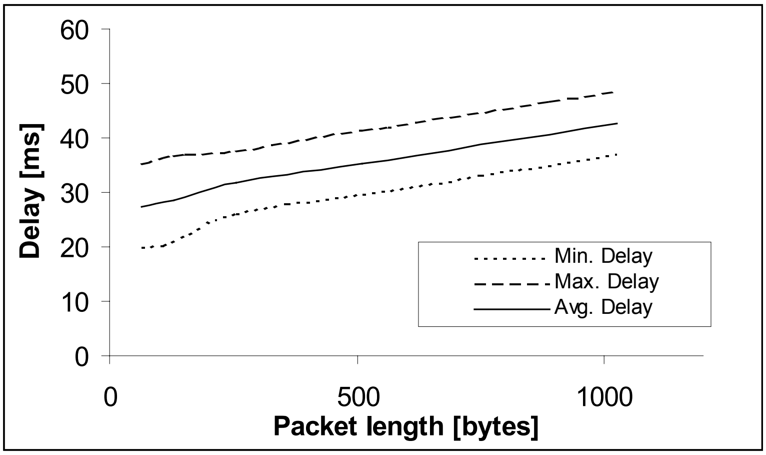


Fig. 7. One-way delay vs. TCL-1 packet length

5.2 Results for TCL-3

The TCL-3 was mainly defined for effective support of TCP-controlled flows. The objective for this service is to guarantee a minimum throughput to the TCP flows. The measurement results reported in this section were obtained assuming that the traffic generated by a number of TCP greedy sources (between PC1 and PC3 – see Fig.5) is submitted to the class in question. The background traffic of CBR type is submitted to other network services with the maximum possible rate corresponding to the assigned capacity for given service. In this way the capacity available for the test TCP connections is limited to the capacity allocated to TCL-3. In the considered case this capacity is 600 kbps with target utilisation factor equal to 0.9 what gives 540 kbps. Notice that the out-of-profile packets in PMM service are not dropped. Consequently, the considered greedy TCP flows can achieve higher rate than requested by the SBR parameters.

The goodput characteristics as a function of best effort traffic for the case of four TCP flows with different declared (and policed) SBR values are depicted on Fig.8. The assigned SBR values were as follows: 135, 135, 70 and 200 kbps. One can observed that the available capacity for TCL-3 is effectively used by the considered TCP connections. When the background traffic exceeds a threshold (in this case 700 kbps) the available capacity is limited to 600 kbps (the capacity dedicated to this service). Moreover this capacity is shared between TCP flows in proportion to their declared SBR rates. For example, the connections with SBR=200 gets 215 kbps while the connection with SBR=70 gets only 95 kbps.

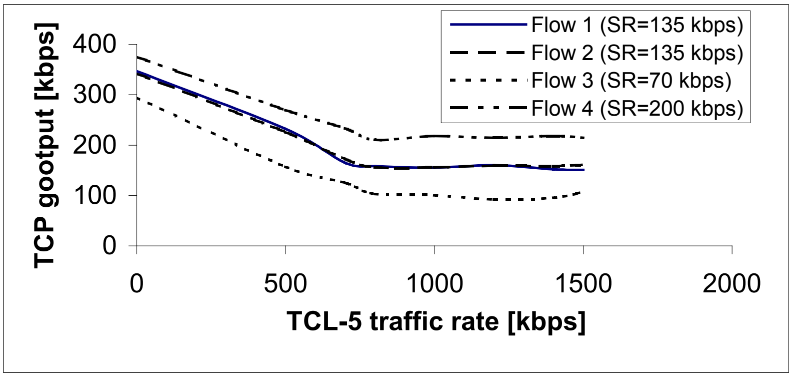


Fig. 8. TCP goodput vs. TCL-5 traffic rate

5.3 Impact of TCL-1 on TCL-3

The impact of TCL-1 traffic on TCL-3 is illustrated on Fig. 9 showing the goodput characteristics of TCP flow 1 and 3 as a function of TCL-1 traffic load. One can observe that increasing the TCL-1 traffic load above the admission limit (133 kbps) decreases the capacity available for TCL-3 and, as a consequence the goodput of TCP flows can be less than the requested rates (SBR values). In the considered case, this effects occurs when TCL-1 traffic is about 230 kbps. This is caused by two factors: (1) there are 100 kbps not assigned to any network service and (2) the target utilisation for TCL-3 is 0.9.

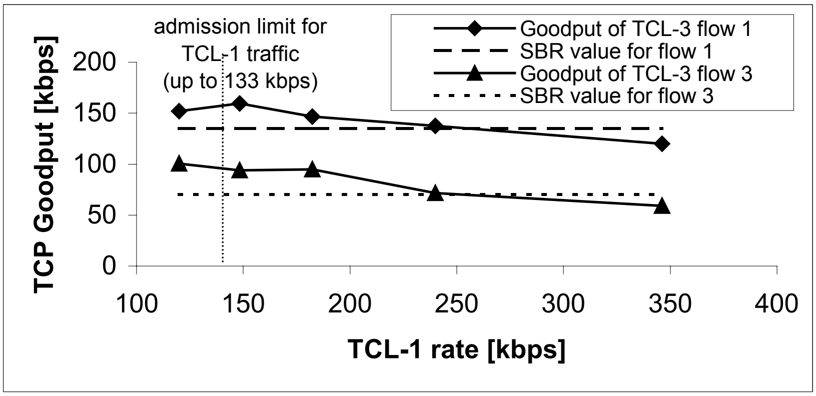


Fig. 9. TCP goodput of flows 1&3 vs. TCL-1 traffic rate

6 Conclusions and Future Work

In the paper the traffic handling mechanisms forced in the AQUILA QoS IP network were described. They correspond to different time scales (from milliseconds to hours or days). The included preliminary measurement results obtained in the AQUILA testbed confirm that by adding new functionalities into the existing IP network there is a possibility to define a set of traffic classes offering differentiated QoS. These experiments were mainly focused on the evaluation of QoS corresponding to the packet and flow level, and were provided for two representative traffic classes (TCL-1 and TCL-3) designated for different types of traffic, non-reactive and reactive.

The following conclusions can be outlined:

- The isolation between different traffic classes can be effectively provided by scheduling mechanisms implemented in the routers.
- The reactive and non-reactive traffic should be definitively submitted to different traffic classes for meeting assumed QoS objectives.
- Different traffic classes require different traffic characterisations and different admission control algorithms.
- The assumed admission control algorithms for the tested traffic classes work according to the expectations.

The currently ongoing work in AQUILA mainly focuses on further trials on the other TCLs (TCL-2 and TCL-4), and on the evaluation of the traffic handling mechanisms at aggregate level (e.g. performances of the Resource Pool dynamics, handling of Reservation Requests).

Future work, for the second phase of the AQUILA project, is to consider inter-domain aspects and to include measurements in the resource control in order to have a feedback from the network performance into the provisioning and resource distribution phase.

References

1. S. Blake et al., “An Architecture for Differentiated Services”, Internet RFC 2475, December 1998.
2. B. Davie et al., “An Expedited Forwarding PHB”, Internet Draft, draft-ietf-diffserv-rfc2598bis-01.txt, April 2001.
3. Y. Bernet et al., A Conceptual Model for DiffServ Routers, INTERNET-DRAFT, draft-ietf-diffserv-model-02.txt, September 2000.
4. Final report COST 242, Broadband network teletraffic: Performance evaluation and design of broadband multiservice networks (J. Roberts, U. Mocci, J. Virtamo eds.), Lectures Notes in Computer Science 1155, Springer 1996.
5. Final Report COST 257, Impact of New Services on the Architecture and Performance of Broadband Networks (COST 257 Management Committee, Phuoc Tran-Gia, Norbert Vicari eds.), ISBN-Nr. 3-930111-10-1, compuTEAM, Wuerzburg, Germany 2000.

6. Deliverble D1201, System architecture and specification for the first trial, AQUILA project consorcium, http//www-st.inf.tu-dresden.de/Aquila/, June 2000.
7. Deliverable D1301, Specification of traffic handling for the first trial, AQUILA project consortium, http//www-st.inf.tu-dresden.de/Aquila/, September 2000.
8. Deliverable D2301, Report on the development of measurement utilities for the first trial, AQUILA project consortium, http//www-st.inf.tu-dresden.de/Aquila/, September 2000.
9. Deliverable D3101, First Trial Integration Report, AQUILA project consortium, http//www-st.inf.tu-dresden.de/Aquila/, March 2001.
10. Deliverable D3201, First Trial Report, AQUILA project consortium, Draft version, April 2001.
11. F. Brichet et al., Stochastic ordering and the notion of negligible CDV, Proc. of 15th International Teletraffic Congress, Washington D.C., USA, 1996.
12. K. Lindberger, Dimensioning and design methods for integrated ATM networks, Proc. of 14th International Teletraffic Congress, Antibes, 1994.
13. A. Elwalid et al., A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node, IEEE Journal on Selected Areas in Communications, p. 1115-1127, 1995

The TCP Control Block Interdependence in Fixed Networks – Some Performance Results

Michael Savorić

Technical University Berlin, Department of Electrical Engineering,
Einsteinufer 25, 10623 Berlin, Germany
savoric@ee.tu-berlin.de
http://www-tkn.ee.tu-berlin.de/~savoric

Abstract. For every TCP connection a TCP sender has a control block in which the current values of the variables and parameters of the connection are stored, e.g., the congestion window size, the slow start threshold, or the (smoothed) round trip time. The information stored in the control blocks of some TCP connections might be useful for other TCP connections, e.g., starting a new TCP connection with more adequate initial values of the variables and parameters than it is designated in standard TCP to increase the throughput of the new TCP connection. This is the idea behind the TCP Control Block Interdependence (TCBI) [1]. Which of the information stored in the control blocks of some TCP connections and how these information are used for a new TCP connection is dependent on the TCBI controller in the TCP sender with TCBI capabilities. In this paper the performance of two different TCBI controllers compared to the standard TCP are investigated by simulations in a fixed network scenario. Since more and more end systems are connected to the internet via wireless LANs, also the influence of packet losses in the last hop of a TCP connection is regarded.

Keywords: TCP Congestion Control, Flow Control, Network Information Reuse

1 Introduction

In the sender of a TCP connection some variables and parameters of the TCP connection are stored in a memory block called TCP control block. Some of these variables and parameters, e.g., the congestion window size, the slow start threshold, or the (smoothed) round trip time, are very important for the accurate and protocol-conform working of the TCP sender. Together these above mentioned and other variables hold the information obtained by the network so far on a per-connection basis (see Fig. 1). During the lifetime of a TCP connection this information is dynamically varying.

It might be – at least – an interesting idea to use the stored network information in the TCP control blocks of some TCP connections for other TCP connections to improve their performance. This is the idea behind the TCP control block interdependence (TCBI) [1].

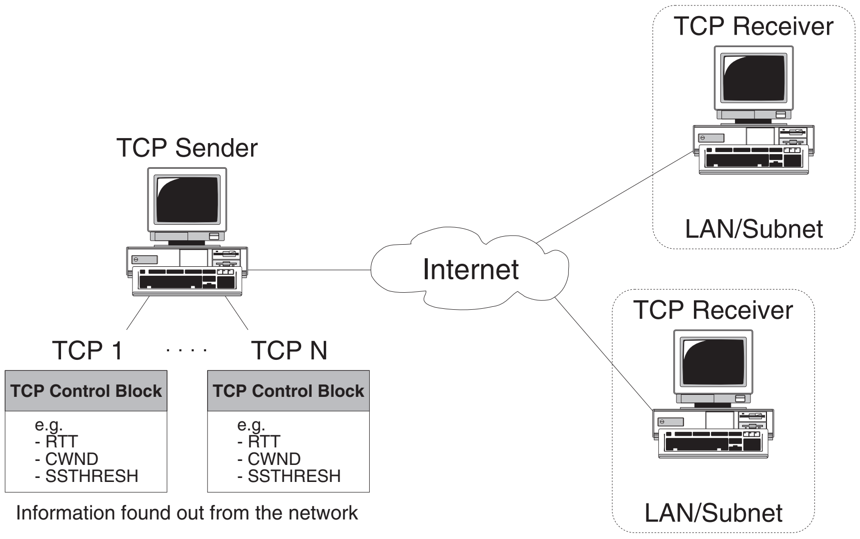


Fig. 1. The TCP control blocks of a TCP sender

Although the TCBI approach has been released four years ago, an accurate investigation on what terms and for which TCP connections the TCBI achieves a performance gain and how large this performance gain is has not been published so far. Based on simulation results this paper gives some answers to these questions.

In this paper a TCP sender with TCBI capabilities is called a TCBI TCP sender or in short notation a TCBI sender. The TCBI information exchange is only reasonable between TCP connections of a TCBI sender which have the same TCP receiver or at least TCP receivers in the same subnet of the network. These TCP connections form a TCBI TCP connection set or in short notation a TCBI connection set.

Which of the information of other TCP connections belonging to the same TCBI connection set and how these information are used for a new TCP connection is dependent on the implementation of the TCBI controller in the TCBI sender.

The TCBI can be used not only for new TCP connections. In a scenario with mobile TCP receivers the TCBI can be used after a handover of a mobile TCP receiver for two reasons: First and similar to the TCBI of a new TCP connection, to increase the throughput of the handover TCP connection by using a higher congestion window size than the current congestion window size — if possible. Second, to reduce the probability of a congestion in the new subpath of the handover TCP connection between the handover switching node in the network and the new base station by using a smaller congestion window size than the current congestion window size — if necessary. The latter case is the more important one from the network point of view, since a congestion negatively influences the throughput of all TCP connections over the congested part of the network.

In this paper the TCBI is only investigated in a fixed network scenario with immobile TCP receivers. In [8] the TCBI is also investigated for handover TCP connections in a mobile network scenario. But the preliminary results based on only a few test simulations cannot be used for a concluding evaluation of the TCBI approach for handover TCP connections. Further research has to be done on this topic.

2 TCP Control Block Interdependence in Fixed Networks

In the following list some thoughts about the feasibility and usability of the TCP control block interdependence in fixed networks are mentioned:

– If the TCP connections of a TCBI sender have different TCP receivers how fine should the granularity be chosen to determine that two or more of these TCP connections have TCP receivers in the same part of the network and can form a TCBI connection set? If the precise segmentation of a class A, B, or C network in subnetworks is not known, only the network part of a class A, B, or C network address can be used for that. But this might be too inaccurate.
– The TCBI can be used not only between a new TCP connection and TCP connections which exist in parallel (ensemble TCBI), but also between a new TCP connection and one or more TCP connections whose lifetimes have recently terminated (temporal TCBI). In some already existing TCP implementations, e.g., in the network part of the Linux kernel, the control block of a terminated TCP connection further exists for a short period of time and can be used for the temporal TCBI. But how long can be the period between a terminated TCP connection and a new TCP connection that the information obtained from the network so far and stored in the control block of the terminated TCP connection is still meaningful and not totally out-of-date and even detrimental for the overall performance of the TCP connections?
  Measurements [5] have shown that a proper observation method for the available bandwidth on a path seen by one TCP connection can be used for other TCP connections on the same path as a fairly good prediction for the available bandwidth up to time periods in the order of tens of minutes.
– The information exchange between TCP connections of a TCBI connection set can be done once, e.g., at the start of a new TCP connection, or dynamically during the whole lifetime of the TCP connections belonging to the same TCBI connection set. This decision can be also depend on the TCP variables for which the TCBI is used, e.g., for all TCP connections of a TCBI connection set only one value for the (smoothed) round trip time or the derived timeout timer seems to be reasonable. Another possibility is to use an aggregate congestion window size and share it (dynamically) over the TCP connections of the TCBI connection set. This is similar to the ideas behind the TCP implementation in the ensemble TCP (E-TCP) approach [3] or in the congestion manager (CM) [6].

- The TCBI might be used to shorten the transient start phase of a new TCP connection with normally smaller congestion window sizes at the beginning of the TCP connection due to the TCP slow start algorithm. Particularly for short TCP connections with only a few segments to send, e.g., for transaction TCP (T/TCP) [2] connections, the TCBI might be very useful, since for a short TCP connection the transient start phase is a large part of the whole lifetime and has a high and negative influence on the overall throughput of this TCP connection.
- Although the TCBI might be very useful for short TCP connections, for an accurate collection of information about the network the TCBI sender should have many TCP connections in parallel or one after another at frequent time intervals and at least some of them should be longer TCP connections with dozens of segments to be transmitted. Only those longer TCP connections have the chance at all to increase their congestion window size to a value that represents the currently available bandwidth in the path to the receiver so that shorter TCP connections can strongly benefit from if they use the TCBI.
- Only those TCP connections can benefit from the TCBI which have a moderate to high bandwidth delay product between the TCBI sender and the TCP receiver and therefore can make use of a higher (initial) congestion window size. Other TCP connections with a small bandwidth delay product, e.g., intra-LAN TCP connections, can not take advantage of a higher (initial) congestion window size.

In general, one TCBI sender rarely has as many TCP connections that the TCBI can be used by expecting a noticeable increase of the mean throughput of all of its TCP connections. But, for example, a highly frequented WWW server or a proxy server with TCBI capabilities have a much higher probability to use the TCBI for increasing the mean throughput of their TCP connections.

3 Simulation Model

The basic structure of the simulated fixed network is shown in the following Fig. 2. The simulation network consists of several TCP senders (S1,..., S3 and BS1,..., BS3) and TCP receivers (R1,..., R3 and BR1, ..., BR3), two routers, and two wired LANs.

The TCP senders S1, S2, and S3 are combined in a TCP sender node with TCBI capabilities between the TCP connections of these three TCP senders, i.e., this TCBI sender node is a kind of a huge server like a proxy server. The other TCP senders BS1, BS2, and BS3 are responsible for the background traffic in the simulation model.

All the receivers BR1, BR2, and BR3 of the background TCP connections and the receivers R1, R2, and R3 of the TCBI sender's TCP connections are stationary.

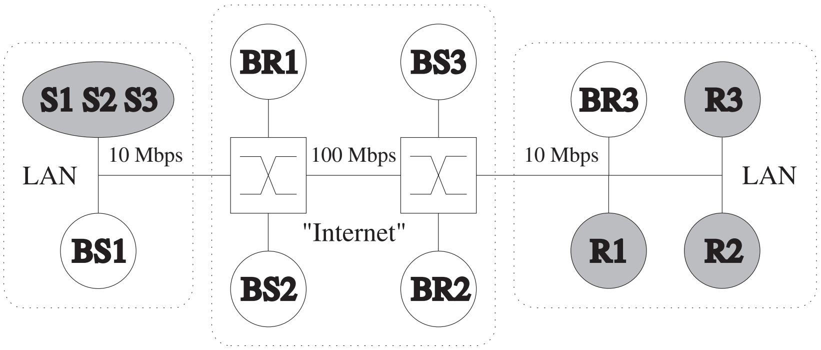


Fig. 2. structure of the simulated fixed network

The routers are connected via links with a bit rate of 100 Mbps and a propagation delay of 10 ms. For each link the routers have a queueing capacity of 20 IP packets.

The background traffic TCP senders BS2 and BS3 are connected to the network via links with a bit rate of 100 Mbps and a propagation delay of 0.25 ms. The background traffic TCP sender BS1 and the TCBI sender are connected to the network via the sender LAN in which the 802.3 transmission protocol is used. The TCBI TCP receivers R1, R2, and R3 and the background TCP receiver BR3 are connected to the network via the receiver LAN in which also the 802.3 transmission protocol is used. Both LANs have a bit rate of 10 Mbps and a propagation delay of 0.0005 ms. The propagation delay values of the LANs are derived from the normal dimension of a LAN and the speed of the signals in a LAN.

The packet loss rate in the receiver LAN is adjustable to find out the influence of different packet loss probabilities in the last hop of a TCP connection on the overall throughput of the considered TCP connections.

In the simulation model two different TCP traffic classes are used: the first class comprises short TCP connections with five segments to send and the second class includes TCP connections whose number of segments to send is coming from a WWW traffic model [7]. This traffic model is derived from real HTTP traces in corporal and educational environments and uses three abstraction levels: The session level, the page level, and the packet level. In this simulation model a simplified version of this WWW traffic model is used which consists only of the first two levels. In every WWW session a lognormal distributed number of WWW pages with pareto distributed page sizes are sent. The time between the pages, i.e., the inter connection time, is gamma distributed. With the exponentially distributed session interarrival time the load in the network can be easily adjusted.

The inter connection time of TCP connections coming from the first class of TCP connections is also gamma distributed.

The TCBI sender has TCP connections from both connection classes: one short TCP connection and two WWW TCP connections. All the background traffic TCP connections are coming from the second connection class.

The whole simulation model is implemented in ns-2 (version 2.1b6) and for all TCP connections the ns-implementation of the TCP NewReno sender is used.[1]

4 Simulation Scenarios

The simulation model is used to investigate two different simulation scenarios: Simulations with a reliable receiver LAN or an unreliable receiver LAN like a wireless LAN. In the simulation scenario with a reliable receiver LAN no packet losses occur in the receiver LAN. In the simulation scenario with an unreliable receiver LAN in the receiver LAN packets are lost with a given packet loss rate (PLR). In addition, due to the background traffic also some packets can be lost in the routers of the network.

With these simulation scenarios the influence of the TCP control block interdependence on the throughput of the TCP connections in fixed networks with different packet loss properties in the last hop can be investigated.

5 TCBI Controllers

Two different variants of a TCBI controller are used to compare the standard TCP in a TCP sender with the ensemble TCBI approach in a TCBI sender. At this stage of the investigation both TCBI controller variants consider only the congestion window size of the TCP connections in the TCBI connection set. The first TCBI controller has an effect only on the initial congestion window size of a new TCP connection whereas the second TCBI controller influences the current congestion window sizes of all TCP connections in a TCBI connection set.

5.1 Mean Value TCBI Controller (MV-TCBI)

This TCBI controller computes the mean of the current congestion window sizes of the already existing TCP connections of a TCBI connection set and assigns this value as the initial congestion window size to the new TCP connection. If no other TCP connection is in the TCBI connection set of a new TCP connection then the congestion window size of the new TCP connection is set to the standard value 1. Of all TCP connections of the TCBI connection set only the congestion window size of the new TCP connection is changed.

5.2 Fair Share TCBI Controller (FS-TCBI)

This TCBI controller computes the sum of all current congestion window sizes of the already existing TCP connections of the TCBI connection set plus 1. This value is used to calculate a congestion window size fair share for all TCP

[1] The ns-implementation of the TCP NewReno sender and the TCP receiver does not perform the connection setup/teardown of a TCP connection. In future investigations of the TCBI approach also the influence of the TCP connection setup/teardown protocol mechanism on the overall throughput of a TCP connection will be considered.

connections in the TCBI connection set. At the beginning of a new TCP connection all TCP connections of a TCBI connection set get this congestion window size fair share as their new congestion window size.

The FS-TCBI controller is less aggressive to the network than the MV-TCBI controller, since with the FS-TCBI controller after a start of a new TCP connection the sum of all congestion window sizes of the TCP connections in the TCBI connection set is only one segment higher than the sum of all congestion window sizes of the TCP connections in the TCBI connection set before the new TCP connection has started. In addition, the FS-TCBI controller is fair to all TCP connections belonging to the same TCBI connection set.

6 Evaluation Metric

For each simulation the mean throughput and the mean initial congestion window size of new TCP connections are compared between the standard TCP and the TCBI controllers. However, only those TCP connections are considered in the mean throughput and mean initial congestion window size computation shown in the following tables which either use the TCBI or do not use the TCBI but could use it, since at least one parallel TCP connection to the same LAN is already established and useful information found out from the network is available. This evaluation metric is used, since in general the share of TCBI TCP connections on all TCP connections depends on the type of the TCBI sender. For example, the many TCP connections of a huge WWW or a proxy server have a higher probability using the TCBI approach than the few TCP connections of an ordinary end system. So, the chosen evaluation metric shows the performance of the TCBI approach independent of the type of the TCBI sender. The overall performance of the TCBI approach on the throughput of all TCP connections can then be easily computed by using the share of the TCBI TCP connections on all TCP connections of the considered TCBI sender.

Two different computations of the mean throughput of the considered TCP connections are regarded in the compare between the standard TCP and the TCBI controllers. The first computation of the mean throughput ($\overline{\mathrm{TP}}_1$) divides the number of sent segments of all considered TCP connections by the duration of these TCP connections. In the second computation of the mean throughput ($\overline{\mathrm{TP}}_2$) for each considered TCP connection a mean throughput is calculated. All these mean throughput values are used to compute the overall mean throughput of the considered TCP connections by a normal arithmetic mean calculation independent on the number of segments sent by the TCP connections. The latter calculation gives a connection-oriented mean throughput which can be understand as the mean throughput a single TCP connection of one of the two TCP traffic classes can expect.

7 Statistical Evaluation Method

For the two simulated scenarios the standard TCP (no TCBI) case is statistically compared with the MV-TCBI case as well as with the FS-TCBI case.

The statistical evaluation method used for this compare is called the t-test for unpaired observations of two alternatives and is described in detail in [10]. The main idea of this method is to compute a confidence interval for the difference of the mean values of both alternatives for a given confidence level. Then the decision criterion is:

- If the confidence interval includes zero, then the two alternatives are not different.
- If the confidence interval is below zero, then the second alternative is the better one.
- If the confidence interval is above zero, then the first alternative is the better one.

Tests with confidence intervals give not only a yes-no answer like other hypothesis tests, they also give an answer to the question how precise the decision is. A narrow confidence interval indicates that the precision of the decision is high whereas a wide confidence interval indicates that the precision of the decision is rather low.

This t-test for unpaired observations of two alternatives is used for the statistical evaluation of the simulation results for the connection-oriented mean throughput($\overline{\mathrm{TP}}_2$) of the standard TCP and both the MV-TCBI controller and the FS-TCBI controller.

In the simulation model together with the used traffic characteristics it is a rare event to have TCP connections to the same receiver or to receivers in the same LAN in parallel and use the TCBI approach. Therefore, the calculated confidence intervals are rather wide. However, in most of the simulations significant differences between the standard TCP and the MV-TCBI or FS-TCBI controller are observable.

In all tables which show the statistical evaluation results of the simulations for each confidence interval also the confidence level (0.90, 0.95 or 0.99) is denoted.

8 Simulation Results

The simulation results for the fixed network and either a relibale or an unreliable last hop with a packet loss rate of 5 % are shown in the following two subsections.

In every table of the simulation results TCP 1 is the short TCP connection and TCP 2 and TCP 3 are the WWW TCP connections of the TCBI sender. Both stated mean throughput metrics of the considered TCP connections are measured in TCP segments per second whereby in every TCP segment the payload length is set to 1000 bytes. In addition, also the mean initial congestion window size ($\overline{\mathrm{cwnd}}$) of the considered TCP connections is shown.

For both simulation scenarios the three different TCP controllers (standard TCP/no TCBI, MV-TCBI, FS-TCBI) are simulated for a simulation time of 250000 s at each case. In this simulation time approximately 23000 TCP connections starting at the TCBI sender can be observed. Only some of them could use or use the ensemble TCBI. In the simulation model together with the chosen traffic load model the possibility to have TCP connections in parallel and use the ensemble TCBI for a new TCP connection is relatively low. An average computation over all simulations shows that approximately 1.5 % of the new TCP connections in the reliable last hop scenario and approximately 4.3 % of the new TCP connections in the unreliable last hop scenario could use or use the ensemble TCP control block interdependence. During one simulation approximately 23000 TCP connections starting at the background traffic TCP senders can be observed.

8.1 Scenario 1: Ensemble TCBI with a Reliable Last Hop

The simulation results for the connection-oriented mean throughput ($\overline{\text{TP}}_2$) are given in the following table 1. They show that for new short TCP connections the MV-TCBI controller has a mean throughput gain of approximately 83 %. Also the new WWW TCP connections benefit from the MV-TCBI controller with a mean throughput gain of approximately 12–15 %. As expected the mean throughput gain for the less aggressive FS-TCBI controller is lower. But a mean throughput gain of approximately 60 % for new short TCP connections is nevertheless considerable. For new WWW TCP connections a mean throughput gain of approximately 5–11 % can be observed.

Table 1. Simulation results for scenario 1

		no TCBI	MV-TCBI	FS-TCBI
$\overline{\text{TP}}_1$ of a new TCP connection [segments/second]	TCP 1	28.86	58.07	53.84
	TCP 2	132.11	156.53	144.49
	TCP 3	142.50	166.25	152.24
$\overline{\text{TP}}_2$ of a new TCP connection [segments/second]	TCP 1	35.99	65.76	57.55
	TCP 2	132.10	151.37	147.19
	TCP 3	139.61	155.93	146.07
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	1.00	16.57	8.76
	TCP 2	1.00	12.32	8.59
	TCP 3	1.00	12.49	8.09

In the simulation scenarios with the TCBI controllers for the sum of all background TCP connections a mean throughput loss of less than 1 % is observed. Thus, in the simulation model and with the observed probability of using the ensemble TCBI in the TCBI sender the background TCP connections are not realy negatively affected by the TCBI TCP connections.

The following Table 2 shows the significance of the simulation results dependent on the chosen confidence level $1 - \alpha$.

Table 2. statistical evaluation of the simulation results for scenario 1

		no TCBI ↔ MV-TCBI	no TCBI ↔ FS-TCBI
$\overline{\text{TP}}_2$ of a new TCP connection $[1-\alpha : \text{conf}()]$	TCP 1	$0.99 : (-35.98, -23.56)$	$0.99 : (-22.80, -12.90)$
	TCP 2	$0.99 : (-35.03, -\ 3.51)$	$0.95 : (-27.56, -\ 2.62)$
	TCP 3	$0.99 : (-32.60, -\ 0.04)$	$0.90 : (-16.87, +\ 3.95)$

8.2 Scenario 2: Ensemble TCBI with an Unreliable Last Hop

The simulation results in Table 3 for the connection-oriented mean throughput ($\overline{\text{TP}}_2$) show that for new short TCP connections the MV-TCBI controller has a mean throughput gain of approximately 72 %. Also the new WWW TCP connections benefit from the MV-TCBI controller with a mean throughput gain of approximately 27–33 %. For the new short TCP connections the FS-TCBI controller has a mean throughput gain of approximately 31 %. For new WWW TCP connections a mean throughput gain of approximately 8–12 % can be observed.

Table 3. simulation results for scenario 2

		no TCBI	MV-TCBI	FS-TCBI
$\overline{\text{TP}}_1$ of a new TCP connection [segments/second]	TCP 1	22.71	26.59	24.90
	TCP 2	68.72	88.37	69.73
	TCP 3	69.95	86.60	78.98
$\overline{\text{TP}}_2$ of a new TCP connection [segments/second]	TCP 1	44.23	76.15	58.08
	TCP 2	96.17	122.25	103.85
	TCP 3	93.55	124.67	105.42
$\overline{\text{cwnd}}$ of a new TCP connection [segments]	TCP 1	1.00	6.01	2.94
	TCP 2	1.00	4.95	2.71
	TCP 3	1.00	5.14	2.59

In the simulation scenarios with the TCBI controllers for the sum of all background TCP connections a mean throughput loss of less than 1 % is observed. Thus, in the simulation model and with the observed probability of using the ensemble TCBI in the TCBI sender the background TCP connections are not realy negatively affected by the TCBI TCP connections.

The following Table 4 shows the significance of the simulation results dependent on the chosen confidence level $1 - \alpha$.

Table 4. statistical evaluation of the simulation results for scenario 2

		no TCBI $\leftrightarrow$ MV-TCBI	no TCBI $\leftrightarrow$ FS-TCBI
$\overline{\mathrm{TP}}_2$ of a new	TCP 1	$0.99 : (-38.20, -25.64)$	$0.99 : (-18.94, -\ 8.76)$
TCP connection	TCP 2	$0.99 : (-37.85, -14.32)$	$0.90 : (-14.91, -\ 0.45)$
$[1 - \alpha : \mathrm{conf}()]$	TCP 3	$0.99 : (-42.42, -19.82)$	$0.99 : (-23.03, -\ 0.71)$

8.3 Summary

Based on the two simulation scenarios the following statements can be made:

– For new short TCP connections the MV-TCBI controller has a considerable throughput gain in the range of 72 % up to 83 % compared to the standard TCP. Also the new WWW TCP connections benefit from the MV-TCBI controller.
– For new short TCP connections the FS-TCBI controller yields a throughput gain in the range of 31 % up to 60 % compared to the standard TCP. This lower throughput gain compared to the MV-TCBI controller is an expected result, since the FS-TCBI controller is less aggressive to the network than the MV-TCBI controller. Also new WWW TCP connections benefit from the FS-TCBI controller.
– In the simulation scenario with an unreliable last hop the throughput difference between the standard TCP and the MV-TCBI or FS-TCBI controller is smaller. This expected result has two reasons: The initial congestion window size for a TCBI TCP connection in the case of an unreliable last hop is smaller than in the case of a reliable last hop, since also the parallel TCP connections are affected on packet losses and therefore have a smaller mean congestion window size. And the first packet loss of a TCBI TCP connection reduces its congestion window size to the standard value and nearly eliminates the performance improvement which can be otherwise expected from a higher initial congestion window size.
– It is remarkable that in the simulation scenario with an unreliable last hop for short TCP connections the connection-oriented mean throughput is higher than in the simulation scenario with a reliable last hop. This has two reasons: First, most of the WWW TCP connections are affected by packet losses in the last hop. Due to the TCP congestion control algorithms these WWW TCP connections reduce their overall sending rate and allocate less bandwidth than in the simulation scenario with a reliable last hop. The now available free bandwidth in the LANs and in the network can be used by the short TCP connections to increase their own sending rate. This effect can compensate and even invert the negative influence on the mean throughput of the lower initial congestion window size of the short TCBI TCP connections in the simulation scenario with an unreliable last hop. Second, since only a few short TCP connections are affected by packet losses in the last hop

most short TCP connections benefit from this available free bandwidth and have a higher mean throughput. Therefore, the connection-oriented mean throughput of the short TCP connections can be higher.

9 Conclusion and Outlook

The simulation results show that the TCBI approach is a promising alternative to the standard TCP. New short TCP connections mostly benefit from the TCBI approach. But also new WWW TCP connections have a remarkable higher throughput compared to the standard TCP.

These simulation results are in accordance with some measurement results obtained in a student's project in which the ensemble and temporal TCBI was implemented in a current linux kernel [9].

At present the two TCBI controllers have only an influence on the congestion window size of a new TCP connection or the already existing TCP connections in a TCBI connection set. In future investigations also other TCP variables and parameters, e.g., the slow start threshold or the (smoothed) round trip time, should be taken into consideration. And the middle-term objective will be the combination of the TCBI approach with a transparent common congestion control of TCP connections belonging to the same TCBI connection set over their whole lifetime. Afterwards, in terms of TCP-friendliness also the sending rate of UDP streams should be controlled with this approach.

References

1. Touch, J.: TCP Control Block Interdependence. RFC 2140 (1997)
2. Braden, R.: T/TCP – TCP Extentions for Transactions. RFC 1644 (1994)
3. Eggert, L., Heidemann, J., Touch, J.: Effects of Ensemble TCP. ACM SIGCOMM Computer Communication Review, Vol. 30, No. 1 (2000) 15–29
4. Schläger, M., Rathke, B., Bodenstein, S., Wolisz, A.: Advocating a Remote Socket Architecture for Internet Access using Wireless LANs. Journal of Mobile Networks and Applications (2001) 23-42
5. Paxson, V.: End-to-End Internet Packet Dynamics. IEEE/ACM Transactions on Networking, Vol.7, No.3 (1999) 277–292
6. Balakrishnan, H., Seshan, S.: The Congestion Manager. Internet Draft draft-ietf-ecm-cm-02.txt, Work-in-Progress (2000)
7. Reyes-Lecuona, A., González-Parada, E., Casilari, E., Casasola, J.C., Diaz-Estrella, A.: A page-oriented WWW traffic model for wireless system simulations. Proceedings ITC 16 (1999) 1271–1280
8. Savorić, M.: The TCP Control Block Interdependence: Some Performance Results. TKN Research Report Series (2001)
9. Daniel, B., Nkweti, P., Wepiwe, G.: Implementing TCP Control Block Interdependence (TCBI) in Linux. TKN Research Report Series (2001)
10. Jain, R.: The Art of Computer Systems Performance Analysis. Wiley & Sons (1991)

Approaches to Support Differentiated Quality of Web Service

Sook-Hyun Ryu, Jae-Young Kim, and James Won-Ki Hong

Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, Korea
{shryu,jay,jwkhong}@postech.ac.kr

Abstract. The exponential rise in the number of Web users has inspired the creation of a diversity of Web applications. Hence, Web Quality of Service (QoS) is an increasingly critical issue in Web services, such as e-commerce, Web hosting, etc. In the future, improved QoS will be linked to a fee for service. Customers expect their requests to be served with a quality proportional to the amount charged to their accounts. Because most Web servers currently process requests on a first-come, first-serve basis, they do not provide differentiated QoS. This paper presents two approaches to implement differentiated quality of Web service. In the user-level approach, the Web server is modified to include a classification process, priority queues and a scheduler. However, with this approach, it is difficult to achieve portability. In this paper, a new, portable user-level approach is presented. In the kernel-level approach, a real-time scheduler to support prioritized user requests has been added to the operating system kernel. Prototype implementations for two approaches have been developed and their performances are evaluated.

1 Introduction

Today, most World Wide Web (WWW or Web) servers do not provide differentiated service quality to different requests of Web users. The Apache Web server [1], one of the most widely used Web servers, handles incoming requests on a first-come, first-serve basis. All requests correctly received are eventually handled, regardless of the type of request on the Web server. In this situation, premium users (i.e., those who pay more for higher quality of service) cannot be protected from overload in the Web server. Consequently, this general Web server does not provide differentiated QoS.

Recently, Web Quality of Service (QoS) becomes a critical issue in various Web services and has been studied in many ways [2,3,4,5,6,7,8]. With the term QoS, we refer to non-functional requirements, such as performance or availability requirements. QoS requirements are concerned with how an application or service will behave at run-time. QoS requirements may differ for different invocations of a service, based on diverse factors, such as user authentication or time of day.

In this paper, we propose two approaches; a user-level and a kernel-level, to provide differentiated QoS using priority-based scheduling. In the user-level approach, the Apache Web server is modified to include a classification process, priority queues and a scheduler. It is difficult to achieve portability when modifying a specific Web server. Thus, we propose a new user-level approach, which includes the classification and scheduling of user requests. In the kernel-level approach, a Linux kernel is extended to include a real-time scheduler to support prioritized Hypertext Transfer Protocol (HTTP) requests. We evaluate the performance of two prototypes and compare response times, throughput, and error rates for high and low priority requests.

The rest of this paper is organized as follows. Section 2 describes related work and Section 3 presents functional and non-functional requirements for design and implementation issues. Section 4 presents the system design architecture and classification approaches and then the performance evaluation results are explained in Section 5. Finally, Section 6 summarizes our work and discusses possible future work.

2 Related Work

A general architecture for Web server QoS was previously outlined in [2]. HP WebQoS is an enhancement to the HP-UX operating environment, providing a unique and advanced platform to assure high service quality for e-services on HP 9000 Enterprise Servers. HP's WebQoS stabilizes service delivery, assuring fast and consistent service quality to customers even under the competitive environment found on the Internet. WebQoS optimizes resources and permits developers to build more cost-effective solutions. WebQoS also enables businesses to prioritize service levels to allow higher service quality for the most important users and applications. The goal was to manage peaks in client request rates and to support differentiated QoS for users. Their solution essentially entails scheduling and admission control to improve the performance of high priority requests. Their architecture includes a management component so that configuration parameters can be remotely set and the server's operation monitored. This closely resembles our work. However, there are several differences. First, changes to the main Apache server code were required for both request classification and scheduling. Our new user-level prototype performs similar tasks without modifying the main Apache code. Second, they do not have the kernel-level approach concept.

Vasiliou et al. [3] presented the design of a QoS architecture that can be added to the Apache Web server to allow the server to provide a differentiated QoS. The QoS module can support changing scheduling algorithms and the values of parameters that are used by the scheduling algorithms. However, the concept of kernel-level approach was not considered. They proposed the services needed to provide a differentiated QoS to clients of a Web site, based on the client's identity and attributes. They also integrated an implementation of the services with the Apache Web server and described a service that can be used to create algorithms to suit a specific company's goal.

Almeida et al. [4] investigated a method to provide a differentiated QoS: priority-based scheduling. The main metric for the QoS is latency in handling the HTTP request to the Web page. They showed both a user-level and a kernel-level approach.

However, the user-level approach can be used only in a specific Web server, and this approach does not support portability. Scheduling policies can be preemptive or non-preemptive. They have implemented preemptive scheduling at the kernel-level, and non-preemptive scheduling at the user-level. Two important aspects of the scheduling policy must be mentioned. First, upon receiving a request, the scheduling policy must decide whether to process the request immediately, or to postpone the execution (sleep policy). Secondly, the scheduling policy must also decide when a postponed request must be allowed to continue (wakeup policy). This work allows a request to continue only in place of a completed request. When a request is completed, the scheduling policy must decide which of the postponed requests, if any, should be selected to execute in its place. If the policy allows lower priority requests to execute in the absence of higher priority requests, the scheduling policy is said to be work-conserving. Otherwise, it is said to be non-work-conserving. A work-conserving policy tries not to allow processes to block while there are waiting requests. In this work, the Sleep and Wakeup policies have been implemented, by using thresholds for the maximum number of requests that can be concurrently handled at each priority level. Thus, a fixed number of slots exist for each priority level, and each incoming request must either occupy a slot (executes), or wait in a queue until it is allowed to execute (blocks). The related work has been compared and summarized in Table 1.

3 Requirements

In this section, we discuss requirements that must be considered during design and implementation. Requirements are divided into two parts – functional requirements and non-functional requirements.

3.1 Functional Requirements

In order to design and implement approaches to provide differentiated quality of Web service, certain functional requirements must be considered. The differentiated Web server requires the following functions: classification of requests, scheduling, and execution of requests.

Table 1. Comparison of Related Work with Our Work

	WebQoS (HP)	Vasiliou et al.	Almeida et al.	Our Work
User-level approach	O	O	O	O
Kernel-Level approach	X	X	O	O
Portability	O	X	X	O
Service levels	3	2	2	2
URL classification	O	O	O	O
Client IP classification	O	O	X	O
User authentication classification	X	X	X	O

The basic role of a general Web server is processing HTTP packets on a first-come, first-serve basis. The general Web server listens to a signal which indicates that a connection has been made. After accepting the connection, the server must serve the user request. After serving the request, the server awaits the next signal.

Unlike the general Web server, the differentiated Web server must provide differentiated QoS using priority-based scheduling. Therefore, we modify the general Web server into the differentiated Web server to add components that support differentiated QoS.

After accepting the connection, the differentiated Web server classifies the user request. The classification of requests is the first concern in the implementation of a differentiated Web server. The classification methods are as various as can be imagined. For example, in a server-based method, we can categorize a Universal Resource Locator (URL) required by the user. In a client-based method, we can classify the user request with the client IP address. After classification of the user request, the server must save it into a priority queue. A scheduler that employs a specific scheduling method assigns the user request in priority queues. The scheduling method is the second concern in implementing a differentiated Web server. As a specific scheduling method is applied, the throughput remains constant during peak demand. In addition, the error rate also remains constant during peak demand. After scheduling the user request, the server must serve the user request. Finally, this server is ready to receive another connection.

3.2 Non-functional Requirements

To deliver differentiated QoS, a Web server can be modified to include a classification process, priority queues and a scheduler. However, portability is difficult to achieve when modifying a specific Web server. A module that supports a differentiated QoS must use a variety of general Web servers, such as Apache Web server and Microsoft's IIS Web server [9]. In addition, a differentiated Web server must be portable on various operating systems. Portability is the major concern among non-functional requirements.

General Web servers have evolved toward a multi-threaded architecture that either dedicates a separate thread to each incoming connection, or uses a thread pool to handle a set of connections with a smaller number of threads. Since a greater number of threads are used to handle user requests, more CPU capacity and memory usage is needed in general Web servers. Further, this condition is applied to a differentiated Web server equally. Because modules are added to support differentiated QoS, more CPU capacity and memory usage is needed in a differentiated Web server. Most Web servers do not serve every user request during peak demand or run short of CPU capacity and memory volume. If both Web servers have an equal CPU capacity and memory volume, the differentiated Web server must use as little CPU and memory usage as possible. Therefore, resource requirements are a secondary concern in non-functional requirements.

4 Design of Two Approaches

In this section, we discuss two approaches towards differentiated QoS in Web services, and present a differentiated Web server architecture and process structures.

4.1 User-Level Approach

Here, we present a user-level approach to support differentiated quality of Web service. In the user-level approach, the general Web server, such as Apache Web server, is only modified to include components for supporting differentiated QoS. These components consist of connection and classification processes, priority queues, a schedule process, and execution processes, as illustrated in Fig. 1. Such components as classification processes, priority queues and the schedule process do not exist in a general Web server.

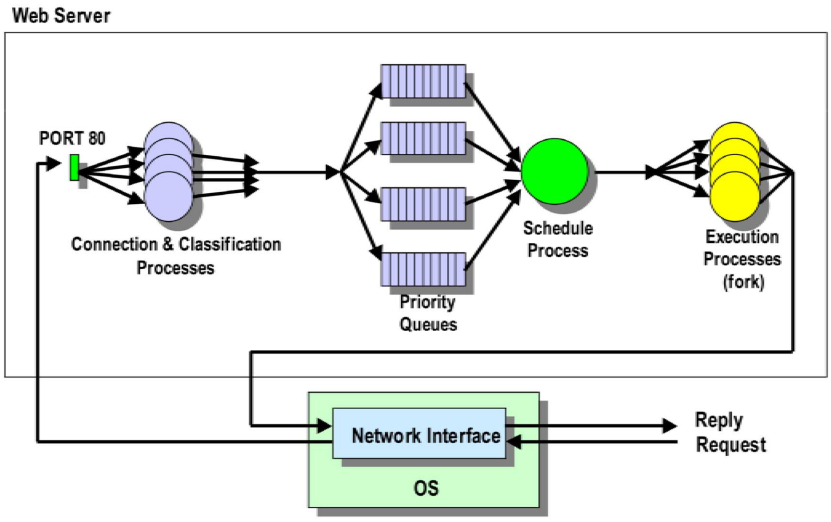


Fig. 1. User-Level Approach

First, incoming user requests from the network interface in the operating system are received through port 80 by connection and classification processes. Port 80 is a well-known port for supporting Web service. These processes classify the requests and place the requests on the appropriate priority queues. Several methods are used to classify requests. These classification mechanisms can be divided into two categories, a server-based and a client-based approach. In a later section, we explain the classification method in detail.

The number of priority queues implies the number of differentiation levels. A priority level is assigned for each priority queue. After requests are classified, the server must realize different service levels for each class of requests. This is accomplished by selecting the order of request execution.

A schedule process selects the next request, based on the scheduling policy. For example, requests from the highest priority queue will be processed first. Execution processes forked by the schedule process may be able to execute requests from any class, and will run until completion. Finally, execution results are sent to the user through a network interface.

Since we must modify a general Web server source code to support differentiated QoS in this approach, the system using this approach cannot handle the large variety of general Web servers. Because of this, we propose a new user-level approach to support portability, as follows.

4.2 A New User-Level Approach for Supporting Portability

Since the user-level approach described in Section 4.1 can be used only in a specific Web server, this approach does not support portability. For this reason, we propose a new user-level approach to support portability.

The new user-level approach consists of a differentiate module and a general Web server, as illustrated in Fig. 2. The differentiate module is an independent program for the realization of a differentiated QoS. The components of the differentiate module are similar to that of the user-level approach described in Section 4.1. However, we do not the general Web server, such as the Apache Web server.

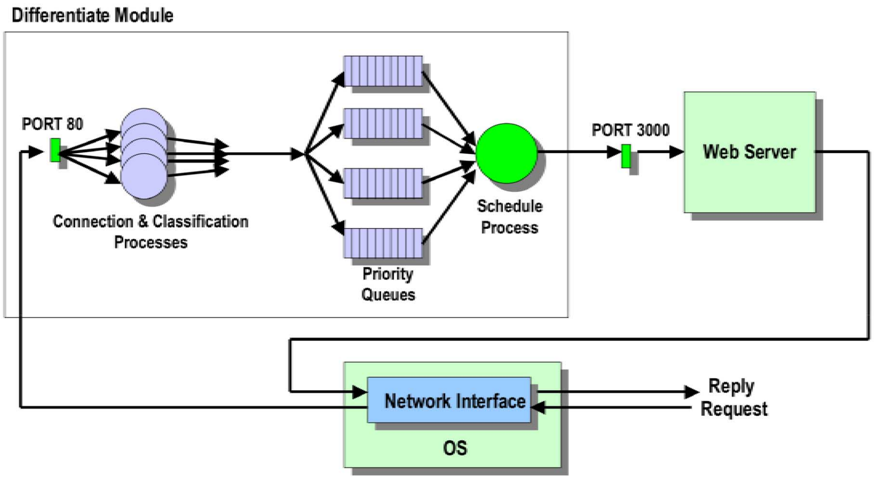


Fig. 2. New User-Level Approach

A schedule process selects the next request based on the scheduling policy. A selected request by a schedule process is sent to a general Web server through a specific port, such as 3000. To accomplish this, the Web service port in a general Web server is configured into a specific port number, except for port 80. The selected request is processed in a general Web server. Finally, the execution result is sent to the user through a network interface.

4.3 Kernel-Level Approach

We considered it heuristically advisable to attempt a kernel-level approach, since processes which act directly on the priorities assigned to the HTTP request might be more effective in controlling executions. Therefore, we use direct mapping from the user-level request priority to a kernel-level process priority. The kernel-level approach is based on the instrumentation of both a general Web server modification and an operating system with a real-time module, as illustrated in Fig. 3.

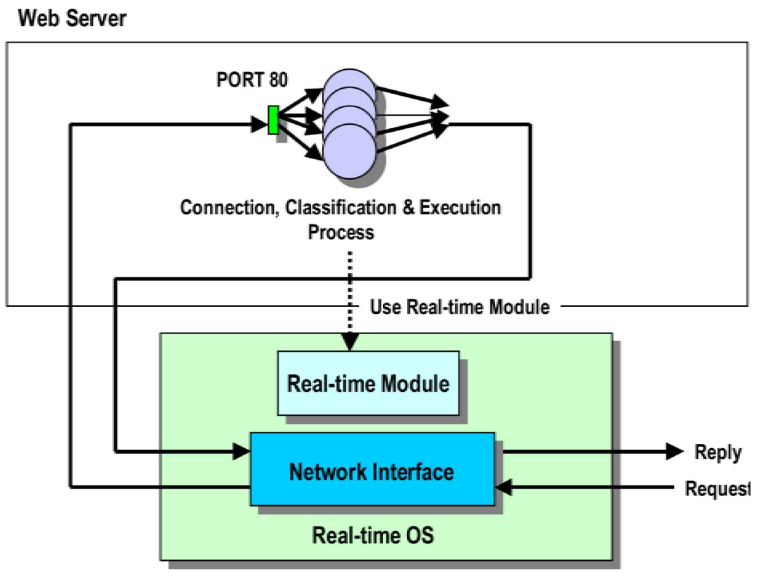


Fig. 3. Kernel-Level Approach

This approach to support differentiated QoS consists of a modification of a Web server to support request classification and an operating system with a real-time module. A general Web server is modified to have each HTTP process call the kernel to record the priority of the request currently being handled. The kernel is responsible for mapping this priority into the process priority. The kernel scheduler decides which process should use the CPU next. The kernel must keep track of all processes currently using the priority, along with their current state.

4.4 Classification Methods

A key requirement to support differentiated QoS is the ability to identify and classify the incoming requests according to each service class. Classification mechanisms can be divided into two general categories: a server-based and a client-based. The server-based method classifies requests according to the contents or destination of the request. The client-based method characterizes requests according to the source of the request.

In the server-based method, the contents of URL of Web requests are used for determining a service level. A URL consists of a protocol header, a server address, a directory name, a file name and an extension. This information can be used to classify the relative importance of the request. Contents can be classified into different priority levels. Destination IP addresses can be used by a server when the server supports co-hosting of multiple destinations (Web sites) on the same node. The server-based method is useful when performance of accessing some part of Web contents in a Web server needs to be protected and isolated from requests to other parts. Typical examples of this content discrimination can be found at various E-commerce sites. Web pages for processing customers' orders are more important and should be responded more reliable than pages for viewing a catalog of products. Further, users do not need any additional processing overhead or advance knowledge of the method.

In the client-based method, there are two kinds of choices for classification. The client's IP address is used to distinguish privileged clients from the non-privileged. This method is the simplest one to implement. However, since the client's IP address might be changed due to proxies or firewalls, this method has limitations. User authentication with username and password is another method used to classify a request to overcome the limitations. Both methods require advance agreements between the differentiated Web server and clients for the service level and quality. However, the client-based methods are able to provide fine-grained Web QoS for different individual users since the Web server can distinguish one client from another. This kind of differentiation is useful in enterprise environments where different users have different roles and priorities.

4.5 Priority-Driven Scheduling Methods in the User-Level Approach

After the requests are classified according to one of the above-mentioned classification schemes and admitted by the classifier, the server must actually realize different service levels for each class of requests. This is done by selecting the order of request execution. Execution processes are autonomous and select requests to process based on the scheduling policy. The scheduling policy may depend on queue lengths. Execution processes may be able to execute requests from any class. Alternatively, to reserve a capacity for higher-class processes, they may be restricted to executing higher-class traffic. The following presents several possible policy guidelines.

- Strict priority – This policy schedules all higher-class requests before lower-class requests, even when low-priority requests are waiting.
- Weighted priority – This policy schedules a class based on its weight importance. For example, one class will receive twice as many scheduled requests if its class weight is twice that of another.

- Shared capacity – This policy schedules each class to a set capacity and any unused capacity can be given to another class. The class may also have a minimum reserve capacity that cannot be assigned to another class.
- Fixed capacity – This policy schedules each class to a fixed capacity that cannot be shared with another class.
- Earliest deadline first – This policy schedules based on the deadline for completion of each request. This can be used to provide a guaranteed predicted response time.

In our work, we use a strict priority scheduling method to provide differentiated Web service because it is very simple and requires less CPU capacity and memory volume than other priority scheduling methods. This condition is sufficient for non-functional requirements. If we use other priority scheduling methods, an additional resource management mechanism will be required.

5 Prototypes and Performance Comparison

In this section two kinds of prototype implementation details are described and performance comparison between two prototypes with various request loads.

5.1 Development Environment

We have implemented both approaches on the development environment, as summarized in Table 2. Obviously, the development environment of the kernel-level approach is similar to that of new user-level approach, except that it uses the Montavista real-time scheduler [10] to support process priority on the operating system kernel level. The Montavista real-time scheduler is a specific scheduler to support a soft real-time kernel.

Table 2. Development Environment

	New user-level approach	Kernel-level approach
OS	Linux kernel 2.2.14	Linux kernel 2.2.14
Realtime kernel	None	Soft realtime kernel (Montavista scheduler)
Web server	Apache 1.3.12	Apache 1.3.12
Language	C, PHP	C, PHP

We have implemented prototypes of these approaches on the Linux kernel 2.2.14 and the Apache Web server 1.3.12. We used C language for programming the prototypes of these approaches. Further, PHP [11] script was used to configure both the classification policy and user information in these prototypes.

5.2 Performance Evaluation

We evaluate the performance of two prototypes. We compare response times, throughput, and error rates for premium and basic client running with priority scheduling. The performance measurements are for high and low priority clients that monotonically increase with each client issuing an equal number of requests. Also, these clients are configured to issue requests simultaneously. Recall that the performance metrics are the throughput, the response time and the error rate of a request as perceived by the server. The proportion of high to low priority requests is 1:1. In Fig. 4, 5 and 6, the high priority requests have a better throughput, response time and somewhat better error rates in both user-level and kernel-level approaches.

In Fig. 4, we show the throughput when we vary the request rate for the user-level and kernel-level approaches. As the offered load increases, reply rate increases linearly until request rate reaches about 90 requests/s in the high priority requests using kernel-level approach. Further, if the request rate is over 90 requests/s, the reply rate is nearly fixed at 90 replies/s. However, as seen in Fig. 4, reply rate increases linearly until request rate reaches approximately 75 requests/s in the high priority requests using a user-level approach. If the request rate exceeds 75 requests/s, the reply rate is nearly fixed at 75 replies/s. This status occurs in low priority requests using both user-level and kernel-level approaches, too. For the most part, the throughput for requests using the kernel-level approach is greater than for requests using user-level approach in both high and low priory requests.

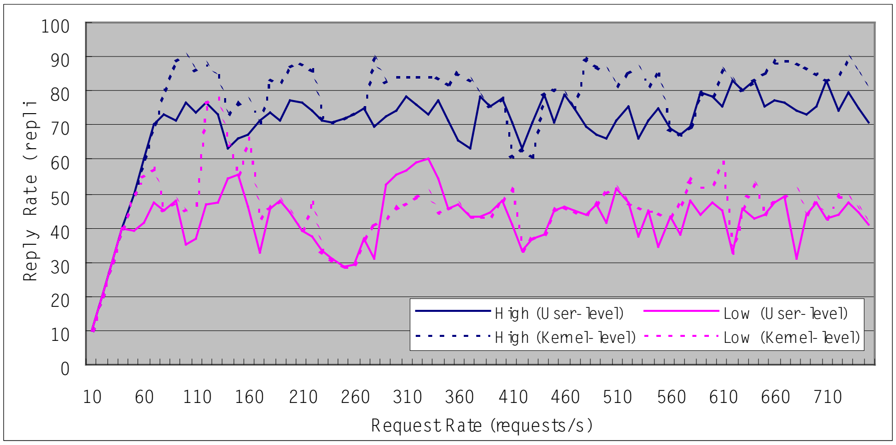


Fig. 4. Throughput in Differentiated Web Server

The response time is optimistic since it does not include requests which are rejected or which encounter an error. The second graph, Fig. 5, plots the average response time in milliseconds for all client requests completed successfully as a function of the total offered client demand rate. The differentiated Web server which uses the user-level approach is longer than one which uses the kernel-level approach in the response time. A new module for supporting portability in the differentiated Web server causes many processing delays.

Finally, the third graph, Fig. 6, plots client requests that encounter an error as a function of the total offered client demand rate in both the user-level and the kernel-level approaches. The error rate for requests using user-level approach is nearly equal to one using kernel-level approach in both high-level and low-level priority requests.

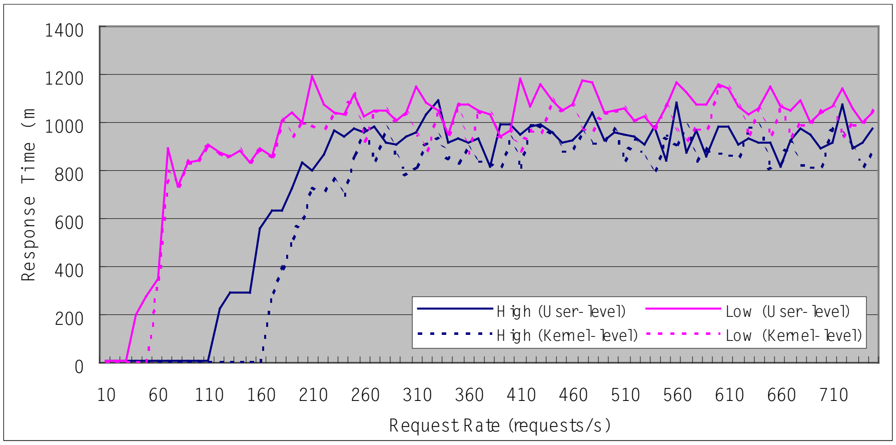


Fig. 5. Response Time in Differentiated Web Server

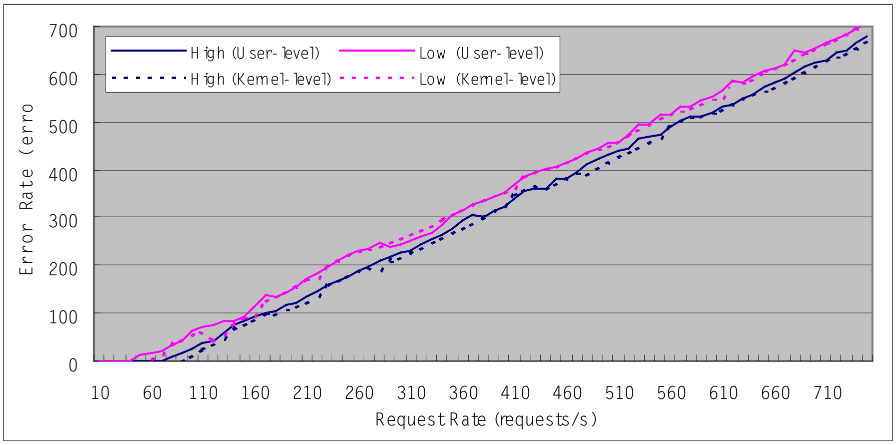


Fig. 6. Error Rate in Differentiated Web Server

Overall, a differentiated Web server that applies a user-level approach becomes lower than one using a kernel-level approach in performance. However, supporting portability in such a user-level approach is an important factor. Most Web server administrators wish that the facilities to support differentiated service are applicable to various Web server and operating systems. It is likely that, with the increasing complexity of Web page design, the differentiation between high and low priority requests will increase as well. In the performance evaluation of the differentiated Web server, the responses of high priority requests are occasionally slower than one of low priority requests. In our view, the drop rate of low priority requests will be faster than one of high priority requests. In this situation, the length of the low priority queue will be temporarily shorter than that of the high priority queue. So this problem results. Future work will be devoted to ascertaining the exact cause of the problem.

6 Conclusion and Future Work

In this paper, we presented two approaches to support Web Quality of Service (QoS) that allows a general Web server to provide a differentiated QoS. Our methods categorize HTTP requests into classes based on classification methods, with the requests of each class handled differently by the desired scheduling method.

High priority clients are given better throughput, response time, and slightly better error rates. Supporting portability in that using user-level approach is considered factor as important. Most Web server administrators intend to implement a facility for supporting differentiated service in a variety of Web servers and operating systems.

We evaluate the performance of our implementations. They include analyzing errors, throughput, and response time of a server that handles one high priority client and one low priority clients with request rates that monotonically increase. This experiment models a situation where there are a various number of clients with subscriptions for high and low quality services. Further, we are making comparative performance evaluations of two approaches supporting differentiated quality of Web services.

More work is needed on scheduling algorithms. Many approaches can be taken to perform the scheduling of requests. The open-queue system of the Web will require complex algorithms to balance processing of requests in each class while maximizing system utilization. Designing a Web server framework to support server QoS is also a complex task. The Web server framework consists of an admission controller, a resource manager, a disk scheduler and a request scheduler. More work is also needed to integrate a server QoS with a network QoS. It is necessary to map their QoS parameters. Finally, we will extend current work on the Web server to other Internet servers, such as FTP server, VOD server, RealAudio server, and so on.

References

1. Apache HTTP Server Project Homepage, http://httpd.apache.org.
2. N. Bhatti and R. Friedrich., "Web Server Support for Tiered Services," IEEE Network, September/October 1999, pp. 64-71.
3. N. Vasiliou and H. Lutfiyya., "Providing a Differentiated Quality of Service in a World Wide Web Server," Proc. of the Performance and Architecture of Web Servers Workshop, Santa Clara, California USA, June 2000, pp. 14-20.
4. J. Almeida, M. Dabu, A. Manikutty, and P. Cai., "Providing Differentiated Levels of Service in Web Content Hosting," Proc. of the Workshop on Internet Server Performance, Madison, Wisconsin USA, March 1998, pp. 91-102.

5. M. Banatre, V. Issarny, F. Leleu, and B. Charpiot., "Providing Quality of Service over the Web: A Newspaper-based Approach," Proc. of the Sixth International World Wide Web Conference, Santa Clara, California USA, April 1997, Paper 149-Tec 110.
6. K. Beyer, M. Livny, and R. Ramakrishnan., "Protecting the Quality of Service of Existing Information Systems," Proc. of the Cooperative Information Systems, 1998, pp. 74-83.
7. R. Pandey, J. Barnes, and R. Olsson., "Supporting Quality of Service in HTTP Servers," Proc. of the SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Puerto Vallarta, Mexico , June 1998, pp. 247-256.
8. T. Abdelzaher and N. Bhatti, "Web Server QoS Management by Adaptive Content Delivery," Proc. of the 7th International Workshop on Quality of Service, London, England, June 1999, pp. 216-225.
9. Microsoft TechNet, http://www.microsoft.com/technet/iis.
10. Montavista Software, http://www.montavista.com.
11. PHP Project Homepage, http://www.php.net.

Understanding the Long-Term Self-Similarity of Internet Traffic*

Steve Uhlig and Olivier Bonaventure

Infonet group, University of Namur, Belgium
{suhlig,obonaventure}@info.fundp.ac.be
http://www.infonet.fundp.ac.be

Abstract. This paper analyzes the characteristics of Internet traffic by studying a six days long trace of the entire interdomain traffic received by an ISP. Our study shows that this traffic is self-similar at time-scales spanning minutes to hours. We show that this self-similarity could be explained by two factors. First, the traffic volume received from each external source exhibits a heavy-tailed distribution. Second, the number of these external sources is also self-similar. Finally, we show that self-similar traffic can be simulated by users transferring exponentially distributed traffic provided that the number of users is self-similar.

1 Introduction

It is now a long time since self-similarity has been uncovered in data networks. From Ethernet traffic [LTW$^+$94] to wide-area traffic [PF95], passing through web traffic [CB96], all have been characterized by statistical self-similarity. Some have modeled quite successfully such traffic with ON/OFF traffic sources [WPT98] [TWS97] while others have tried to fit a Markov-modulated model [RL97]. Whether or not one can model such high variability or partially explain its causes, the main fact remains the corresponding burstiness and its related problems, from packet handling mechanisms to link capacity planning. While telephone networks have successfully relied on statistical multiplexing for reducing operational costs, the Internet has this unfortunate feature that limits traffic aggregation from reducing traffic variability, also called traffic self-similarity. [PKC97] has discussed some implications of self-similarity on network performance as well as the impact of network handling mechanisms on traffic characteristics. While packet handling mechanisms may in part be taken as responsible for short-range traffic self-similarity, it is unlikely that such short-term aspects like packet-level dynamics can explain large time-scale variability as shown in [CB96]. In that respect, the purpose of this paper is to better understand the long-term traffic self-similarity, in the order of minutes and hours.

More precisely, we study the traffic self-similarity by partitioning network traffic dynamics into its probabilistic and dynamic aspects. An important idea

* This work was partially supported by the European Commission within the IST ATRIUM project.

has already been raised in [CB96] and [PKC96], in that the heavy-tailed characteristics of the objects' sizes to be transferred over the network could suffice for generating self-similarity. It has also been shown that heavy-tailed file transfer duration and file sizes can lead to high variability.

We show in this paper that there are two very different parts of the network traffic generation process that have different implications on self-similarity. On one side, a heavy-tailed objects sizes distribution suffices for network traffic self-similarity to arise. On the other side, the dynamics of the number of IP sources sending traffic during a particular time interval allows for the distributional properties to actually generate a self-similar traffic pattern.

The remainder of this paper is structured as follows. Section 2 presents the context in which the traffic traces were gathered. We introduce self-similarity in section 3 by studying the evolution of the total traffic. We then look at traffic components in section 4 to find plausible causes for this self-similarity. We first look at the the heavy-tailed properties of the traffic trace in section 4.1 and explain its role in self-similarity. We go on by studying the evolution of the number of traffic sources in section 4.2 and try to assess in section 4.3 which part between heavy-tails and traffic dynamics is more likely to cause self-similarity.

2 Traffic Statistics

Many researchers have chosen to study the behavior of network traffic by relying on packet level traces from a particular link (see [LTW$^+$94] [PF95] [MC00] among others). Such traces allow the analysis of the traffic at the packet scale, but require a large storage space. For this reason, most of the used traces either correspond to a low traffic volume or a short period of time. In this paper, we have taken a different approach. In order to better understand the long-term behavior of the traffic, we consider a less precise trace that spans six complete days. For this, we rely on a `Netflow` [Cis99] trace collected at the border routers of the Belgian research ISP Belnet during December 1999 [UB00]. The trace covers the interdomain traffic received by the ISP on all its access links and accounts for 2.1 Tbytes of traffic. The studied ISP provides access to the Internet as well as to high speed European research networks to universities, government and research institutions in Belgium. At that time, its national network was based on a 34 Mbps backbone linking major Belgian universities. Its users are mainly researchers or students with direct high speed connections to the 34 Mbps backbone, although some institutions also provide dial-up service to their users. It was also at that time the ISP with the largest capacity in Belgium.

This network is connected to a few tens of external networks with high bandwidth links. It maintains high bandwidth peerings with two transit ISPs, the Dutch SURFNET network and is part of the TEN-155 European research network, without providing any transit service to its peers. In addition, the ISP is present with high bandwidth links at the Belgian and Dutch national interconnection points with a total of about 40 peering agreements in operation. The `Netflow` trace we used aggregated the information from all upstream links, in a manner that we have the incoming traffic information like if there was only one access link to the local ISP.

`Netflow` provides us with the aggregated information of the layer-4 flows, by recording the starting time, the ending time and the total volume in bytes for each unidirectional TCP and UDP flow. The utilization of `Netflow` forces us to approximate the layer-4 flows as equivalent to fluid flows. More precisely, a flow transmitting M bytes between T_{start} and T_{stop} is modeled as a fluid flow transmitting $M/(T_{stop} - T_{start})$ bytes every second between T_{start} and T_{stop}. This approximation obviously leads to an inaccurate estimation of the short-term burstiness of the traffic but allows for longer traffic traces collection. The time granularity of the trace is one minute, i.e. all traffic volume information is summarized over equally-spaced one minute intervals throughout the six days. The part of the `Netflow` statistics on which we rely throughout this paper is the information of the total traffic volume received from every external IP address for every minute of the measurements. We thus have for each minute the information of the number of IP sources that are sending traffic during a particular minute as well as the corresponding traffic volume for each of them. Even if the trace granularity is one-minute, it accounts for more than 42 million values recorded over more than 8600 samples, with an average of 4912 IP addresses sending traffic per minute. Hence it provides a very fine measurement of the traffic dynamics for time-scales spanning minutes to hours.

3 Total Traffic

The focus of this section is on measuring self-similarity of the total traffic time-series. We thus look at total traffic received at the incoming access links for every one-minute time interval during the 6 days of the measurements. Figure 1 shows the evolution of total traffic during the period of the measurements. While the global evolution of total traffic exhibits a stable daily periodicity, with peak hours located during the day, there are important deviations around the average traffic evolution throughout the day that give self-similarity this highly bursty look, over many time-scales. The mean traffic over the six days was slightly larger than 32 Mbps, with a one-minute maximum peak at 126 Mbps and a standard deviation of 21 Mbps. The trace begins around 1 AM and finishes six days later around 1 AM also.

Statistical *self-similarity* in the context of stationary discrete-time processes is defined through the following procedure. Let $X = \{X_i, i \geq 1\}$ be a stationary sequence, where

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X_i, \quad k = 1, 2, ... \qquad (1)$$

represents the *m-aggregated* sequence obtained by summing the original sequence X over non-over-lapping blocks of length m and averaging over each block. The

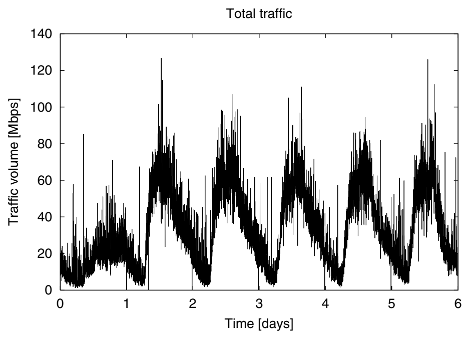


Fig. 1. Total traffic evolution.

sequence $X^{(m)} = \{X_k^{(m)} : k = 1, 2, ...\}$ is said to be *asymptotically self-similar* if

$$X \overset{d}{=} m^{1-H} \; X^{(m)} \quad as \quad m \to \infty \tag{2}$$

where $\overset{d}{=}$ denotes equivalence in distribution. The parameter H (for the Hurst parameter) indicates the degree of long-range dependence (self-similarity) in the time-series. The value of H for self-similar processes is between 0 and 1, with values under 1/2 meaning short-range dependence while values over 1/2 relate to long-range dependence. This paper relies on several estimators for measuring the parameter H, the main measure for self-similarity in time-series. Because H is difficult to measure in practice, we rely on several estimators (see [Ber94] [TT98] [TTW95]) to obtain a gross picture of its value range. We focus on qualitative self-similarity, meaning that the value of H is not overly important, but rather whether it is larger than 1/2 or not. It must be clear however that self-similarity is an asymptotic concept, meaning that statistical inference is difficult on the sole basis of finite measurements.

The first estimator is the well-known R/S statistic [Ber94]. Plotting the R/S statistic for large k on a log-log scale must be scattered around a straight line of slope H, which is found to be around 1 for total traffic (upper left of figure 2).

The second method for measuring H is the *aggregated variance* method, where one computes the sample variances of the *m-aggregated* series around their sample means

$$s^2(m) = \frac{1}{m-1} \sum_{i=1}^{m} X^{(m)}(k) - \bar{X}^{(m)} . \tag{3}$$

Then plot $s^2(m)$ as a function of m on a log-log scale. This should follow for large values of m a straight line with negative slope $2H - 2$. This method gives a value of H very close to 1, as shown on figure 2 (upper right), with a slope around 0.

The third estimator for H relies on the time dependence between the samples, so that long-range dependent data should exhibit very slowly decaying sample correlations proportional to k^{2H-2} for $1/2 < H < 1$. Figure 2 (lower left) shows this very slow decay by plotting the sample autocorrelations as a function of the lag k on a log-log scale. We can see that the decay rate is far slower than $k^{-0.5}$, thus the estimated H is close to 1.

The three previous methods were based on the time-domain, the final one is the periodogram, a frequency-domain based one. If the series exhibits *long-memory* then the estimated spectral density at the origin should behave like

$$f(\lambda) \sim c_f |\lambda|^{1-2H} \ \ as \ \ |\lambda| \to 0. \tag{4}$$

Hence plotting the periodogram near the origin with a log-log scale should roughly follow a straight line of slope $1 - 2H$. This estimator gives a value between 0.75 and 1 in our case (lower right of figure 2).

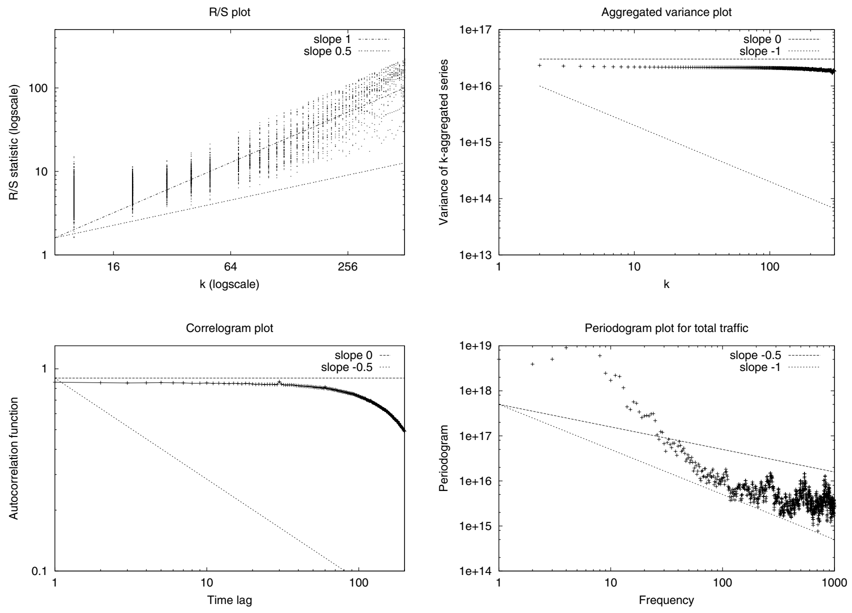


Fig. 2. Estimation of H for total traffic.

It must be noted that most of the estimators for H are defined in an asymptotic way and therefore provide a good estimate for very large datasets only. However, our time-series being quite long, with more than 8600 samples, we can be quite confident with the previous estimators at least in the value range they provided, with H closer to 1 than 1/2 for all of them, meaning that our time-series exhibits a strong self-similarity.

4 Understanding the Long-Term Self-Similarity

Section 3 has described total traffic variability by showing how close to 1 the estimated value of the estimators for H are. In this section, we analyze which parts of the dynamics of the network traffic are due to be responsible for this self-similarity.

4.1 The Role of Heavy-Tails

As already pointed out in [CB96], heavy-tails could play an important role in traffic self-similarity. Traffic variability can be thought as an aggregated random phenomenon generated by the dynamics of user's that are retrieving files over the network. Intuitively, it makes sense that whenever the probability of users transferring a very large amount of bytes does not decay fast enough for large values of the transferred object, this is due to generate large bursts that are persistent at many time-scales. Heavy-tailed distributions conceptualize this idea, in a probabilistic way. For that purpose, this section studies the probability mass of the amount of traffic that is seen for every IP source for every one-minute time interval of the trace.

A random variable X is said to have a *heavy-tailed* distribution if

$$P[X > x] \sim x^{-\alpha}, \; as \; x \to +\infty. \tag{5}$$

A random variable Y is said to be exponentially distributed if

$$P[Y > y] = \alpha e^{-\alpha y}, \; with \; \alpha > 0 \; and \; y > 0. \tag{6}$$

The exponential distribution is associated with *memoryless* processes, that do not depend on the past. Heavy-tailed distributions however characterize *long-memory* processes, with strong time-dependence structures that vanish very slowly. The main difference between heavy-tailed and non-heavy- tailed distributions, in the context of this paper, lies in the asymptotic decay rate of the tail that is not exponential in the case of heavy-tailed distributions [Res97]. This means that when fitting an empirical heavy-tailed distribution, the decay rate of the tail is far slower than exponential, namely like a power function. To illustrate the heavy-tailedness of network traffic, figure 3 shows the probability mass (on a log-log scale) of the amount of traffic seen during one-minute time intervals for individual IP addresses during the measurements. We also plotted two reference power tails, one with $\alpha = 1$ and the other with $\alpha = 2$. This shows that the small values are distributed like a heavy-tail with exponent within $[1, 2]$ while the tail of the distribution happens to be more flat.

Our intent in this paper is not to assess which particular heavy-tailed distribution fits best our data. Thus, we will only shortly discuss the estimation of the exponent α. Figure 4 presents an estimate of α that illustrates the rather strong heavy-tailed characteristics of our trace. Let $X_1 \geq X_2 \geq ... \geq X_n$ denote the ordered statistics of the data, X_1 being the largest value of the data,

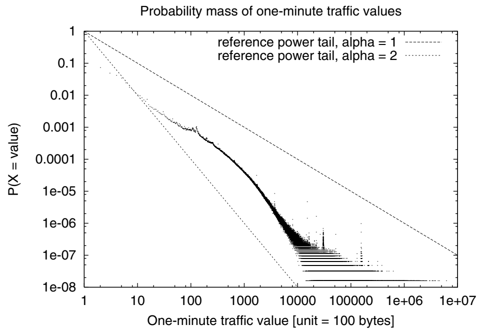


Fig. 3. Heavy-tails in one-minute traffic values.

X_2 the second largest value, and so forth... The Hill estimator [Hil75] gives the estimation of α for the k largest values of the data set and is defined by

$$H_{k,n} = \frac{1}{k} \sum_{i=1}^{k} log(\frac{X_i}{X_{k+1}}) \tag{7}$$

As presented on figure 4, the Hill estimator gives an α well under 1. The thing to notice concerning the Hill plot is the estimate of α for the largest values of the dataset (low order statistics) that remains extremely low, rendering the very slow decay of the tail, hence the strong persistence of large values within traffic sent by each individual IP source during one-minute periods.

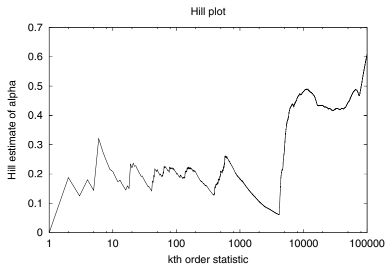


Fig. 4. Estimation of alpha.

4.2 The Role of Traffic Dynamics

Heavy-tails are a distributional property, in the sense that it provides an intuitive explanation of the high variability of total traffic, through the probabilistic persistence of high bursts among the traffic volume sent by individual IP addresses. Self-similarity can also be regarded as a dynamical characteristic of the traffic, related to the extreme variability independent of the time-scale at which one looks at the time series.

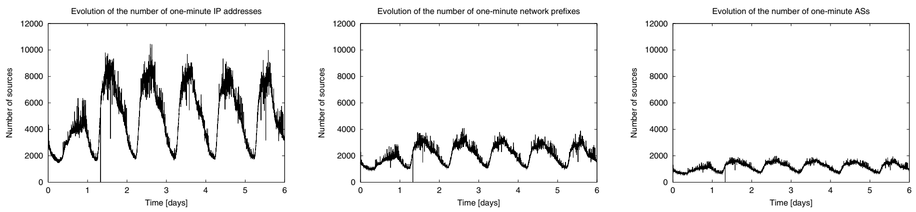


Fig. 5. Evolution of traffic sources: IP addresses (left), network prefixes (middle) and ASs (right).

Besides values distribution that contributes to the traffic volume part of the total traffic sample path, the number of IP addresses that are sending traffic during a particular minute supplies the other part of the traffic generation process, the dynamical one. Using the number of "active IP addresses" over one-minute intervals can be regarded as a broad picture for the number of instantaneous flows. It approximates the real process of the user's (be they humans or machines) communicating over the Internet. Figure 5 presents the evolution of the number of IP addresses (left), network prefixes (middle) and autonomous systems (right) that are sending traffic during a given one-minute interval. A "network prefix" is the aggregation of all IP addresses contained within a domain corresponding to a `<prefix/netmask length>` pair appearing in the BGP routing table of the studied ISP. An "autonomous system" (AS) is the aggregation of all network prefixes contained within a domain corresponding to the destination AS number appearing in the *AS path* information of the BGP routing table of the studied ISP. Figure 5 shows the evolution of the three types of traffic sources with an identical y-axis scale. All traffic sources types exhibit a similar variability, the only difference being their absolute number. The average number of sources over the measurements is 4912 for IP addresses, 2100 for network prefixes and 1176 for ASs. A more detailed study of interdomain flows can be found in [UB00].

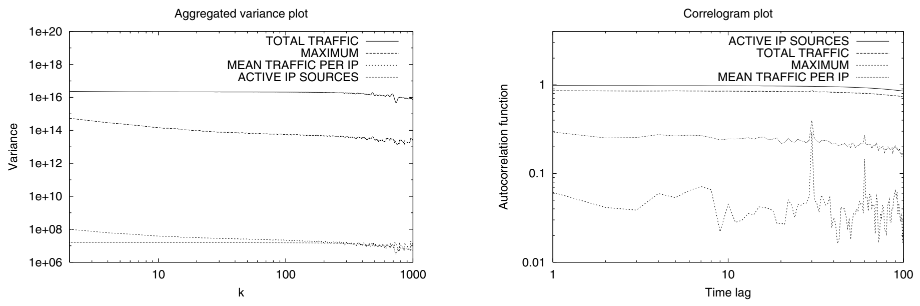


Fig. 6. Comparison of H for traffic components.

As presented on figure 6, the number of "active" IP addresses during one-minute intervals exhibits the same kind of self-similarity than total traffic. Even if it is difficult to assess whether some component of the traffic exhibits a stronger self-similarity than another, looking at the *aggregated variance* and the *correlogram* of some components of the traffic will give us some useful information. The upper part of figure 6 shows the *aggregated variance* plot for the evolutions of: the total traffic, the number of IP sources that are sending traffic during every minute, the mean traffic per IP source and the maximum amount of bytes sent by an IP source during every minute. The lower part of figure 6 shows the *correlogram* for the same four concepts. *Aggregated variance* is almost constant for `IP sources` and `total traffic` while it decreases slightly for the two other concepts. The *correlogram* also indicates a stronger self-similarity for IP sources and total traffic, while the other two concepts have correlations that are close to or below the $2/\sqrt{n}$ confidence limit (around 0.02 for our sample) that prevents us from inferring anything about actual correlations for them. Even if such behavior does not automatically imply that IP sources could better explain self-similarity in the sample path, it constitutes a plausible root. Network prefixes and ASs provide similar results to IP addresses but are not presented due to space limitations.

4.3 Heavy-Tails Revisited

Having studied two possible causes for self-similarity in the traffic cannot tell which one is the most important, if such ever exists. It has already been shown [CB96] that heavy-tails in the distributional properties of the transferred objects was a sufficient condition for generating self-similar traffic under a wide range of conditions. The purpose of this section is to better show the role of heavy-tails for what concerns self-similar traffic.

Let us try the following experiment. Assume we can change the distribution of the sizes of the one-minute transfers so that instead of being heavy-tailed, they were exponentially distributed. For that purpose, we use the characteristics of our traffic trace as a basis and modify the one-minute distributions of the values appearing for the IP sources so that they conform to an exponential distribution, but under the constraint that the total simulated traffic be the same as for the original total traffic trace. Exponential distributions make that possible because they are completely defined by their mean, so that given a particular number of IP sources that are active during a given minute, it is possible to distribute the traffic values exponentially so that the simulated total traffic is equal to the one of the traffic trace.

The procedure for generating every one-minute distribution is the following:

1. Determine in the trace the total amount of traffic T_k (in bytes) seen during minute k as well as the number N_k of distinct IP addresses that are sending traffic during that minute.
2. For each minute k, generate an approximation of the exponential distribution with mean T_k/N_k so that the simulated traffic corresponds to a total of

about T_k bytes and a number of points of about N_k points by relying on the exponential distribution formula

$$P(X = x) = \frac{N_k}{T_k}\, e^{-(N_k/T_k)x}. \tag{8}$$

Step 2 requires that we follow a continuous exponential distribution while in fact we need N_k points. Since we want to generate an equivalent of N_k points that accounts for T_k bytes, we generate a discrete exponential distribution that approximates the continuous one. Thus, instead of integrating a continuous curve, we sum an exponentially distributed collection of points so that we have the equivalent of N_k points providing T_k bytes. Generating this discrete set of points requires that we stop summing the discrete exponential at some value, i.e. the higher bound of the given one-minute simulated sample. We choose this point to be $\max_k$ which corresponds to the value x such that $P(X = x) \geq 1/N_k$, thus the highest value of the discrete distribution which should occur if we had actually N_k discrete points occurring in the simulated distribution. The exact value of $\max_k$ is found thanks to

$$\frac{N_k^2}{T_k}\, e^{-(N_k/T_k)\max_k} \geq \frac{1}{N_k}, \tag{9}$$

saying that the value max_k has to occur at least one time amongst N_k. The remaining of the algorithm attributes to each discrete value from 0 to $\max_k$ its frequency of occurrence as well as its traffic volume in bytes, so that the sum of the frequency of occurrence over all generated values sums up to about N_k and the traffic volume to about T_k. Figure 7 presents the pseudo-code for the generation of the simulated distributions.

```
foreach minute $k$ {
    foreach $value$ = 0 to $\max_k$ {
        // Attributing to $value$ its frequency of occurence
        frequency(value) = $(N_k^2/T_k) * e^{-(N_k/T_k)value}$
        // Attributing to $value$ its traffic volume
        volume(value) = $value * (N_k^2/T_k) * e^{-(N_k/T_k)value}$
    }
}
```

Fig. 7. Pseudo-code for generating exponentially distributed values.

The leftmost part of figure 8 shows the probability mass of the simulated exponential distribution for the whole simulation. The simulated distribution is obviously not a perfect exponential one. As can be seen on figure 8, the end of the exponential tail deviates somewhat from its ideal trajectory due to the discrete nature of our simulation.

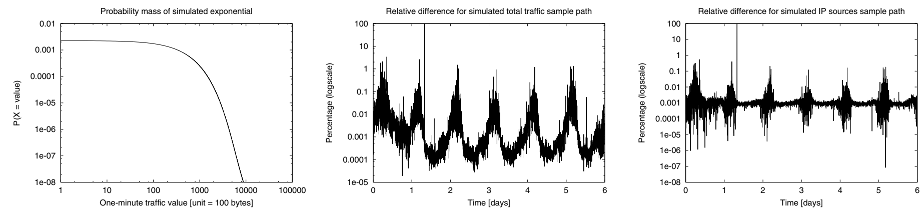


Fig. 8. Simulating exponentially distributed sources: exponential probability mass (left), relative difference in total traffic evolution (middle) and relative difference in IP sources evolution (right).

The other two graphs of figure 8 show the evolution in the "relative difference" between the simulated one-minute total traffic and T_k

$$\mid (T_k - \sum_{value=0}^{\max_k} volume(value))/T_k \mid \tag{10}$$

(middle) and between the simulated number of IP sources and N_k

$$\mid (N_k - \sum_{value=0}^{\max_k} frequency(value))/N_k \mid \tag{11}$$

(right). The relative difference between simulated and original sample paths shows how close the simulation comes to the original, with a global difference smaller than 0.01 % for total volume and smaller than 0.001 % for the number of IP sources. Note that it is possible to better approximate T_k and N_k by using a higher resolution on the values and by cutting the discrete distribution at a larger maximum value.

While rather simple, this simulation shows that heavy-tails in the distribution of traffic values are not alone responsible for generating self-similar traffic on time-scales between minutes and hours. We have shown that by changing the distribution of the amount of traffic IP addresses are sending during any particular minute in a manner that prevents high bursts to occur does not preclude self-similarity to happen in the total traffic sample path. Even if this does not prove that the arrival process of "active" IP sources is responsible for that, we can be sure about the limited role of large bursts (and also heavy-tails) in terms of traffic volume for what concerns self-similarity.

5 Conclusion

In this paper, we have analyzed in details a six days long trace of all the incoming interdomain traffic of an ISP to better understand the long-term self-similarity of Internet traffic. The detailed analysis of the long-term traffic self-similarity has produced three important findings.

First, at this long time-scale, the interdomain traffic received by an ISP appears to be self-similar. This confirms the analysis of shorter traces available in the literature.

Second, we have shown that this self-similarity could be explained by two different elements. The first is the total amount of traffic sent by the external IP addresses that follows a power tail distribution. The second, and this point has been rarely analyzed in the literature, is that the number of sources (i.e. IP addresses) also exhibits a self-similar path.

Third, we have shown that it is possible to simulate self-similar traffic by considering exponentially distributed traffic values and self-similar sources. This indicates that the large bursts have a limited role in traffic self-similarity and that the number of active sources plays an important role in the self-similarity of Internet traffic.

Acknowledgement

This paper would not have been written without the traffic trace provided by Marc Roger from the Belgian research network Belnet.

References

[Ber94] J. Beran. Statistics for Long-Memory Processes. Monographs on Statistics and Applied Probability, Chapman & Hall, 1994.

[CB96] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *SIGMETRICS'96*, pages 160–169, May 1996.

[Cis99] Cisco. NetFlow services and applications. White paper, available from `http://www.cisco.com/warp/public/732/netflow`, 1999.

[Hil75] B. Hill. A simple approach to inference about the tail of a distribution. Annals of Statistics, 3(1975), pages 1163–1174, 1975.

[LTW$^+$94] W. Leland, M. Taqqu, W. Willinger and D. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM* Transactions on Networking, February 1994.

[PF95] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM* Transactions on Networking, 3(3):226-244, June 1995.

[PKC96] K. Park, G. Kim and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In Proc. Fourth International Conference on Network Protocols, October 1996.

[PKC97] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. In Proc. of SPIE International Conference on Performance and Control of Network Systems, November 1997.

[Res97] S. Resnick. Heavy Tail Modeling and Teletraffic Data. Annals of Statistics, 25(1997), pages 1805-1869, 1997.

[Res98] S. Resnick. Why Non-Linearities Can Ruin the Heavy-Tailed Modeler's Day. In "A Practical Guide to Heavy Tails: Statistical Techniques and Applications", Birkhauser, Boston, 1998.

[RL97] S. Robert and J.-Y. Le Boudec. New models for self-similar traffic. Performance Evaluation 30(1-2), pages. 57–68, 1997.

[MC00] S. McCreary and K. Claffy. Trends in wide area IP traffic patterns : a view from Ames Internet Exchange. Available from `http://www.caida.org/outreach/papers/AIX0005/`, 2000.

[TTW95] M. Taqqu, V. Teverovsky and W. Willinger. Estimators for long-range dependence: an empirical study. Fractals, (3):4:785-798, 1995.

[TT98] M. Taqqu and G. Samorodnitsky. On Estimating the Intensity of Long-Range Dependence in Finite and Infinite Variance Time Series. In "A Practical Guide to Heavy Tails: Statistical Techniques and Applications", Birkhauser, Boston, 1998.

[TWS97] M. Taqqu, W. Willinger, and R. Sherman. Proof of a fundamental result in self-similar traffic modeling. *ACM/SIGCOMM* Computer Communications Review, 27(1997), pages 5–23, 1997.

[UB00] S. Uhlig and O. Bonaventure. On the Cost of Using MPLS for Interdomain Traffic. In Proc. of *QOFIS2000*, Berlin, September 2000.

[WPT98] W. Willinger, V. Paxson, and M. Taqqu. Self-similarity and heavy tails: Structural modeling of network Traffic. In "A Practical Guide to Heavy Tails: Statistical Techniques and Applications", Birkhauser Verlag, Boston, 1998.

Network Dimensioning with MPLS[1]

Luís Cardoso

Portugal Telecom Inovação, Rua Eng. José F. Pinto Basto,
3810-106 Aveiro, Portugal
luis-o-cardoso@ptinovacao.pt

Abstract. This paper presents a procedure for MPLS network dimensioning that allows for multi-hour dimensioning of networks supporting simultaneously peer-to-peer and client-server services. The dimensioning model is able to take into account several LSP attributes: degree of survivability (link disjoint and node disjoint cases), maximum hop count, usable colours and preferred routes. The dimensioning problem is a combined capacity design and routing problem where the LSP sets are calculated in order to minimise the network operational costs. This problem is formulated as an integer programming problem, which is solved through an heuristic based on Lagrangean relaxation with sub-gradient optimisation. Results show that the procedure can design networks with realistic size in seconds using a standard PC platform.

1 Introduction

IP networks are currently evolving from their original architecture, capable of supporting a single Best Effort type of service, toward a new advanced architecture characterised by the capability of differentiating a multiplicity of Classes of Services (CoS) providing different levels of performance. This evolution enables the utilisation of IP networks to offering a wide variety of highly valuable services, creating new business opportunity for Telecom operators and accelerating the process of renewing the network infrastructures. To be successful, however, this process requires a greater capability of controlling network performances with respect to the past. As a consequence, there is an increasing interest toward the definition and the implementation of techniques exploitable to meet the desired level of performance under different operational conditions. This new field of activity is usually referred to as Traffic Engineering (TE) for IP networks.

IP TE has been defined as "that aspect of Internet network engineering that deals with the performance evaluation and performance optimisation of operational IP networks" [1] and impacts on several network engineering aspects, such as

- Network planning and dimensioning
- Configuration and control of routing schemes

[1] This document is based on results achieved in a EURESCOM Project; this does not imply that it reflects the common technical position of all the EURESCOM Shareholder/Parties. The author gratefully acknowledges the support of EURESCOM for carrying out this work, and wishes to express a special thank to all P1006 participants (British Telecommunications, Deutsche Telekom, MATAV, CSELT, Telenor, Telefónica).

- Configuration and control of bandwidth management policies
- Configuration and control of congestion control functions

Basically, the TE problem is an optimisation problem [8] having a non-real-time component and a real-time component. The non-real-time component is part of the network planning and dimensioning process and has the objective of optimising the network with respect to the nominal expected traffic, taking into account the operator requirements in terms of Quality of Service (QoS) to be provided and in terms of network reliability and resilience. The real-time component, on the other hand, is part of the network management and control process and has the objective of adapting the network in response to traffic variation or equipment faults so that QoS objectives can still be met. This document focuses mainly on TE aspects related to the network planning and dimensioning phase.

Normal routing currently used in IP networks is hop-by-hop routing based on shortest path calculations. This means Open Shortest Path First (OSPF) or ISIS protocol. This routing scheme is insensitive to the network load and allows little control over the traffic flows. Due to the fact that all the traffic toward a destination is routed to the shortest path, it may happen that this path becomes congested although alternative paths with available resources exist but cannot be used. TE should be able to reduce the congestion hot spots that occur with routing protocols that are insensitive to load by spreading the traffic onto alternative paths. Thus TE may be defined as the ability to move trunks away from the path selected by layer 3 routing and onto a different path. The explicit routing feature of Multi-Protocol Label Switching (MPLS) was introduced to address the shortcomings associated with current IP routing schemes, providing means for ingress routers to control traffic trajectory precisely. Controlling the way in which traffic flows are routed into the network is of fundamental importance for resource optimisation and it is one of the main objectives of TE.

In MPLS, explicit Label Switched Paths (LSPs) can be used to configure different logical networks on top of the physical network. These LSPs can be thought of as virtual trunks that carry flow aggregates generated by classifying the packets arriving at the ingress routers of an MPLS network into Forward Equivalence Classes (FECs). Therefore, a logical network composed by a set of explicit LSPs can be configured in the network to support the traffic flows of each FEC. A similar approach has been used in the past for ATM networks [9], where Virtual Path Connections are used instead of LSPs. This paper extends that work to MPLS networks, and presents a dimensioning procedure that takes into account a set of LSP attributes that act like constraints that must be satisfied when routing each LSP. This work was carried out in DISCMAN project [5].

The organisation of the rest of the paper is as follows. In section 2, we give a short introduction to MPLS. In section 3 the dimensioning procedure is presented. In section 4 we present a case study and in section 5 present some conclusions.

2 Multi-protocol Label Switching

MPLS uses labels in the header to switch the packets, in a similar way to ATM. All packets can be divided into subsets called Forward Equivalence Classes (FEC). The idea is that these subsets are forwarded in the same manner. Classification into FECs is done using packet filters that examine header fields such as source address, destination address, and type-of-service bits. The granularity of the packet forwarding mechanism may therefore vary depending on this classification. The mapping of packets to FECs is only performed once when the packets enter an MPLS domain.

A router that supports MPLS is known as Label Switching Router (LSR).

The MPLS path is called a Label Switched Path (LSP). An explicit LSP is one whose route is determined at the originating node. When we explicitly route an LSP, we call it an LSP tunnel or a traffic-engineering tunnel. LSP tunnels are unidirectional. The source router is called the head-end and the destination router the tail-end. The LSPs can be set up manually or by the use of some protocol. This protocol can e.g. be Resource Reservation Protocol (RSVP) or Label Distribution Protocol (LDP).

Regarding the use of LSPs for different classes of service (CoS) two choices are possible. One choice is to separate LSPs for separate CoS (segregation). But it is also possible to use the 3 bits CoS field in the MPLS label (EXP field). This will make it possible to use up to eight different Differentiated Services (DiffServ) codepoints over a single LSP tunnel by appropriate mapping. In this way different CoS can be integrated on the same LSP. The traffic will then follow the same route, but different EXP value may imply that the packets are treated differently by the LSRs. Specific traffic handling operations, such as queuing algorithms and drop precedence, can therefore still be supported on a CoS basis.

A traffic trunk is a single traffic class that is aggregated into a single LSP. If this LSP contains more than one traffic class, each traffic class represents a traffic trunk. An LSP can thus contain several traffic trunks. The number of trunks within a given topology has a worst case of one traffic trunk per class per pair of edge routers. The MPLS merge functionality is necessary to make this scale; i.e. the use of MPLS merge can reduce the number of trunks substantially. When a new trunk is established it can merge with an existing trunk if they share the same traffic class and from a given point share the remaining explicit route. This is achieved by merging the MPLS paths (many-to-one mapping of labels). In this way trunks with common exit points constitute a single sink tree. The number of such trees will be proportional to the number of edge nodes. The number of trees can be reduced more if trunks for different classes follow the same path. But this may not always be desirable because the ability of LSPs to meet but remain distinct and possibly separate again gives MPLS an additional freedom of route choice that is not available with routing based solely on destination address.

MPLS can operate on a label stack. Operations on this stack are push, pop and swap. This can be used to merge and split traffic streams. The push operation adds a new label at the top of the stack and the pop operation removes one label from the stack. The MPLS stack functionality can be used to aggregate traffic trunks. A common label is added to the stack of labels. The result is an aggregated trunk. When this MPLS path is terminated the result will be a splitting (de-aggregation) of the aggregated trunk into its individual components. Two trunks can be aggregated in this way if they share a portion of their path. MPLS can in this way provide hierarchical forwarding. This may be a very important feature. A consequence may be that the transit provider need not carry global routing information, thus making the MPLS network more stable and scalable than a full-blown routed network [7].

For each MPLS path one or more backup paths can be set up. This path will be activated if the working path becomes unavailable. By this mechanism backup trunks are created. An important property with MPLS is the capability to recover quickly from faults (50 ms) by switching to a backup path. This is much lower than what is possible at IP-layer. At IP-layer the routing protocols must compute alternate paths and information has to be exchanged between the routers. This is not possible to accomplish within 50 ms.

3 MPLS Network Dimensioning

3.1 Introduction

Several features can be considered in the process of network design. For example, it is common sense that existing services do not have the same traffic behaviour during time. For example, business services have higher offered traffic in periods that are complementary to residential services. This fact can lead to optimisation in network utilisation. This is achieved allocating alternatively the network resources to each of the service types in the periods where they require more bandwidth. In order to achieve optimisation gains, it is necessary to consider that the MPLS network has dynamic reconfiguration capabilities. This means that it is possible to reconfigure LSPs in programmable time instants at the management plane of MPLS network. A multi-hour design procedure can lead to significant savings in the overall network cost.

Typically, existing services are either peer-to-peer or client-server based. In peer-to-peer services, there are traffic flows between any pair of nodes with attached users. Thus, an LSP must be configured for each user node pair. In client-server services (e.g., audio-on-demand servers, database access), there are traffic flows between a user node and one of the server nodes of the service. Thus, an LSP must be configured between each user node and a server node.

This paper presents a procedure that performs multi-hour network dimensioning and considers simultaneously mixed peer-to-peer and client-server services. It is assumed that all LSPs to be routed and their attributes are known. In the case of client-server services the attributes are origin node, a set of pre-defined candidate server nodes, maximum bandwidth and, optionally, preferred routes. In the case of peer-to-peer services the attributes are origin node, destination node, maximum bandwidth and, optionally, usable colours, maximum hop count, preferred routes, and survivability. Link attributes must also be defined: maximum utilisation and colour. The attributes defined for each LSP act like constraints that must be satisfied when routing each LSP. The preferred routes attribute is a set of routes from which the LSP route must be selected. The usable colours can be used to forbid a link to be used by a certain LSP, *e.g.*, red links cannot be used for a green LSP. Maximum hop count attribute can be used to limit the maximum number of LSRs crossed for a particular LSP. For the survivability, we consider two types: link disjoint and node disjoint. In both cases, when this attribute is set for a particular LSP, the tool splits the LSP in two, giving each one a bandwidth between 50% and 100% of the original LSP. When node disjoint is considered, the two routes of the LSPs must be node disjoint along the entire path between origin and destination. When link disjoint is considered, the two routes of the LSPs must be link disjoint, but in this case they can have common nodes.

3.2 Problem Formulation

Let the network be represented by an undirected graph (N, A) whose nodes and arcs represent LSRs locations and available transmission facilities between LSR locations. Each element of A is defined by an undirected arc (i,j), with $i, j \in N$. The set of possible interface types to install in any arc is denoted by T and α_t is the bandwidth of interface type $t \in T$. We also define f_{ij} as the colour of the arc (i,j) and u_{ij} as the maximum utilisation. Let Y_{ij}^t denote the maximum number of interfaces of type $t \in T$ that can be installed on (i,j). The operational and maintenance cost associated with the use of one interface of type $t \in T$ in the arc (i,j) is denoted by C_{ij}^t. Let H be the set of time periods h. Each LSP k_h is defined by the origin node $o(k_h)$, the set of possible destination nodes $d(k_h) \in D(k_h)$, the bandwidth in the direction from origin to destination $b(k_h)$, the bandwidth in the direction from destination to origin $\underline{b}(k_h)$, the set of preferred routes $R(k_h)$, the set of usable colours $U(k_h)$, the maximum hop count $m(k_h)$ and the type of survivability $s(k_h)$ (no_survivability, link_disjoint and node_disjoint). Set $D(k_h)$ is (i) a single destination for peer-to-peer services and (ii) the set of server nodes for client-server services. The optimisation model uses the following set of variables. Integer variables y_{ij}^t that define the number of interfaces of type t that are installed on arc $(i,j) \in A$. Route binary variables x_{ij}^{kh}, when equal to one, define that LSP k_h passes through arc $(i,j) \in A$ in the direction from node i to node j. Route binary variables $\underline{x}_{ij}^{kh}$, when equal to one, define that LSP k_h passes through arc $(i,j) \in A$ in the direction from node j to node i. The following integer programming model determines the lowest cost physical network given all LSPs attributes:

$$\text{Minimise} \sum_{(i,j)\in A} \sum_{t\in T} C_{ij}^t y_{ij}^t \tag{1}$$

Subject to:

$$\{ x_{ij}^{kh}, \underline{x}_{ij}^{kh} : x_{ij}^{kh} = 1 \wedge \underline{x}_{ij}^{kh} = 1 \} \text{ is a path subject to constraints set } \{R(k_h), U(k_h),$$

$$m(k_h), s(k_h)\} \text{ from } o(k_h) \text{ to } d(k_h) \in D(k_h),\ k_h \in K_h,\ h \in H \tag{2}$$

$$\sum_{k_h \in K_h} (b(k) \cdot x_{ij}^{kh} + \underline{b}(k) \cdot \underline{x}_{ij}^{kh}) < u_{ij} \sum_{t \in T} \alpha_t y_{ij}^t \text{ , } (i,j) \in \text{A , } h \in \text{H} \tag{3}$$

$$\sum_{k_h \in K_h} (b(k) \cdot \underline{x}_{ij}^{kh} + \underline{b}(k) \cdot x_{ij}^{kh}) < u_{ij} \sum_{t \in T} \alpha_t y_{ij}^t \text{ , } (i,j) \in \text{A, } h \in \text{H} \tag{4}$$

$$y_{ij}^t < Y_{ij}^t \text{ , } (i,j) \in \text{A, } t \in \text{T} \tag{5}$$

$$x_{ij}^{kh} \in \{1,0\}; \ \underline{x}_{ij}^{kh} \in \{1,0\}; \ y_{ij}^t > 0 \text{ and integer} \tag{6}$$

The objective function (1) represents the total cost of the network solution as a function of the number of interfaces of each type installed in each network arc. Constraint (2) forces the solution to be a constrained path from origin to destination for all LSPs to be supported by the network. As it will be understood later, there is no need to explicitly define constraint (2). Constraints (3) and (4) impose that the total bandwidth installed in each arc is enough to support the bandwidth occupied by the LSPs that cross the arc in both directions and in all time periods. Finally, constraint (5) guarantees that the number of interfaces of each type in each arc is not greater then their maximum values.

3.3 Problem Solution

The solution to the optimisation problem is obtained using Lagrangean relaxation with sub-gradient optimisation. Departing from the original problem, a new optimisation problem is obtained by applying Lagrangean relaxation to constraints (3) and (4), which we refer to as the Lagrangean Lower Bound Problem (LLBP). To derive the LLBP, a set of Lagrangean multipliers is introduced, one for each of the relaxed constraints. For any arbitrary set of non-negative Lagrangean multipliers, the solution of LLBP is a lower bound of the original problem [2]. Using the x variables solution of LLBP, it is possible to calculate a feasible solution in the original problem using constraints (3) and (4) to find the minimum values for variables *y*. To compute different sets of Lagrangean multipliers, we use sub-gradient optimisation [6]. This technique is an iterative process that, for a given set of Lagrangean multipliers, calculates another set of multipliers that try to maximise the objective function value of the LLBP. At the end of the procedure, a final solution is obtained which is the best of all calculated feasible solutions. The solution of the LLBP for the route variables is a shortest path calculation in the case of client-server services. For the peer-to-peer services, we developed an algorithm that deals with multiple constraints. In this algorithm, we use the SPLDP (shortest pair of link disjoint paths algorithm) and the SPNDP (shortest pair of node disjoint paths algorithm) presented in [3], which gives the shortest pair of link disjoint and node disjoint paths, respectively. We also use the MDSPHL (modified Dijkstra shortest path hop-limit algorithm) presented in [4], which gives the shortest path that satisfies hop-limit constraint. The algorithm that determines the route for each LSP is briefly described below.

```
if R(k_h)<>∅
  Calculate minimum cost route from R(k_h);
else
{
  bool exist_colour_shortest_path = false;
  if U(k_h)<>∅
  {
    Calculate a new set of arcs A', where all unusable arcs
    are pruned;
    Calculate colour shortest path using Dijkstra(N,A'), if
    path exists set exist_colour_shortest_path = true;
  }
  if s(k_h)<>no_survivability and exist_colour_shortest_path
  {
    bool survivability_exist = false;
    if s(k_h) == link_disjoint
      survivability_exist = SPLDP(o(k_h), d(k_h), N, A');
    else if s(k_h) == node_disjoint
      survivability_exist = SPNDP(o(k_h), d(k_h), N, A');
    if survivability_exist = false
      Use colour shortest path for both LSPs;
  }
  else if s(k_h)<>no_survivability
  {
    bool survivability_exist = false;
    if s(k_h) == link_disjoint
      survivability_exist = SPLDP(o(k_h), d(k_h), N, A);
    else if s(k_h) == node_disjoint
      survivability_exist = SPNDP(o(k_h), d(k_h), N, A);
    if survivability_exist = false
      Dijkstra(o(k_h), d(k_h), N, A);
  }
  else if m(k_h)<>∞ and exist_colour_shortest_path
  {
    bool maximum_hops_path_exist = false;
    maximum_hops_path_exist = MDSPHL(o(k_h), d(k_h), m(k_h), N,
    A');
    if maximum_hops_path_exist = false
      Use colour shortest path;
  }
  else if m(k_h)<>∞
  {
    bool maximum_hops_path_exist = false;
    maximum_hops_path_exist = MDSPHL(o(k_h), d(k_h), m(k_h), N,
    A);
    if maximum_hops_path_exist = false
      Calculate shortest path using (N,A);
  }
  else if exist_colour_shortest_path
    Use colour shortest path;
  else
    Calculate shortest path using (N,A);
}
```

This algorithm is proposed in such a way that when it is not possible to comply with some of the constraints, a solution is selected taking into account only the remaining constraints.

4 Case Study

In order to illustrate the applicability of the dimensioning procedure, we present the following case study. The network topology, shown in Figure 1, has 21 nodes and 46 links.

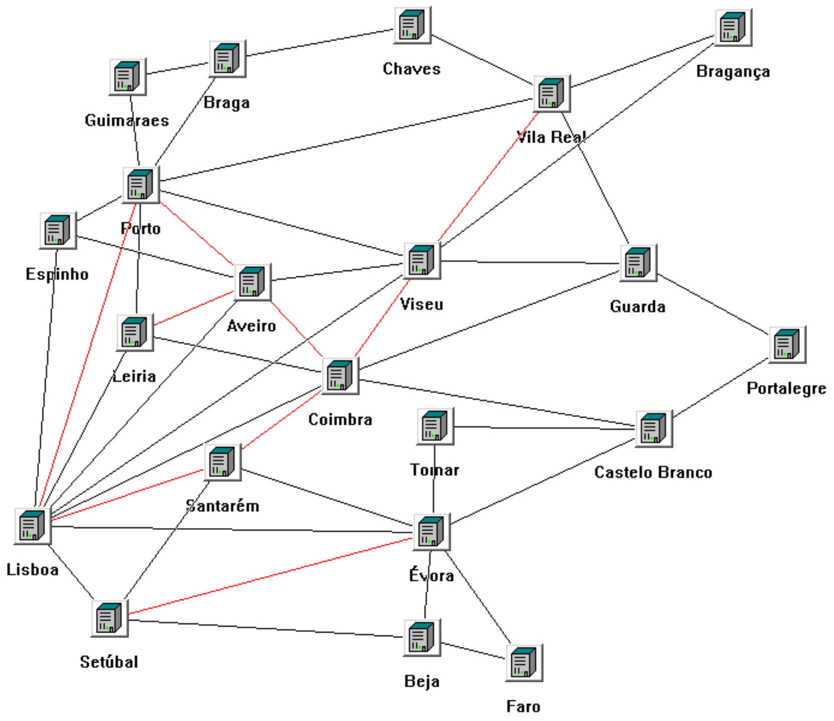


Fig. 1. Network topology of the case study

Three interface types are considered: Channelized E1, Unchannelized E3 and OC3c/STM-1, with the costs defined in Table 1. The final cost of each interface is calculated as 2 × switching cost + transmission cost × link length.

We have considered two services, *Video-telephony (VT)* and *Database Access (DA)*, and two time periods. Each service is defined by the flow bit rate, the set of end user nodes, and the number of flows for each pair of user nodes and for each time period. Service *VT* is a peer-to-peer service, with flow bit rate of 128 kbps. The *DA* service is a client-server service, with flow bit rates of 40 kbps in the client-server direction and 4000 kbps in the server-client direction. Both services have end users at all nodes. There are two servers for the *DA* service located at Porto and Lisboa nodes. The number of flows in each connection was randomly assigned using a uniform distribution between 0 and 100. For each pair of user nodes the LSP bandwidth is calculated as number of flows × flow bit rate of the service.

Table 1. Interface types of the case study

	Bandwidth (Mbps)	Switching Cost	Transmission Cost
Channelized E1	2	100	1
Unchannelized E3	34	1000	10
OC3c/STM-1	155	3000	40

We tested this network in two cases: (i) without constraints and (ii) imposing only link survivability (85% on all LSPs), for uni- and multi-hour design (in the uni-hour case we consider for each LSP the hour with more number of flows). For the client-server service, we considered two cases: (i) fixed server (each user is attached to the closest server) and (ii) selected server (the tool is allowed to select the best server for each node).

The results were as follows:

Table 2. Results of the case study, scenario 1

			Uni-hour		Multi-hour	
			Cost	Time* (sec)	Cost	Time* (sec)
No Constraints	Fixed Server		941,344	11.08	842,589	16.35
	Selected Server		934,849	11.05	836,679	16.55
Survivability	Node Disjoint	Fixed Server	1,128,400	24.47	975,690	42.55
		Selected Server	1,117,280	24.25	973,934	41.96
	Link Disjoint	Fixed Server	1,114,240	23.39	949,513	40.99
		Selected Server	1,107,680	23.00	947,125	41.11

*Using an AMD Athlon 800Mhz with 128 MB RAM

As expected, gains are obtained when adopting a multi-hour approach and a selected server strategy: the cost difference between fixed server/uni-hour and selected server/multi-hour is between 11.1% (no constraints) and 15.0% (imposing a link disjoint survivability). We note that, although the results can vary significantly with the network and traffic scenario, in general considerable gains are obtained by resorting to a multi-hour approach and selected server strategy.

Imposing survivability significantly increased the cost of the achieved solution, and the safest option (node disjoint) is the most expensive. This is because we have considered an 85% level of survivability in the considered cases and, therefore, the network has to accommodate 70% more bandwidth for the peer-to-peer service.

Finally, we created a more complex scenario combining different constraints and setting all links *Black*, except Porto-Lisboa, Porto-Aveiro, Aveiro-Leiria, Aveiro-Coimbra, Lisboa-Santar□m, Santar□m-Coimbra, Viseu-Coimbra, Viseu-Vila Real and Set□bal-□vora, which were set to *Red*.

We have imposed on some of the LSPs a maximum of 6 hops, a node disjoint survivability of 85% and used the colour constraint to forbid them from using red links. The results were as follows:

Table 3. Results of the case study, scenario 2

	Uni-hour		Multi-hour	
	Cost	Time* (sec)	Cost	Time* (sec)
Fixed Server	1,195,890	25.31	1,029,070	43.98
Selected Server	1,095,510	24.66	973,172	44.25

*Using an AMD Athlon 800Mhz with 128 MB RAM

In this case, the costs of the solutions are equivalent to the ones presented previously with survivability constraints. This result illustrates that, among all LSP attributes, the survivability is the one that influences more the cost of the obtained solutions. Comparing the computational times, a similar conclusion can be drawn: the colour and maximum hop count attributes do not impose significant computing time penalties.

5 Conclusions

In this paper we described a procedure that performs MPLS network dimensioning. The dimensioning procedure calculates the routes of explicit LSPs and the interfaces that need to be installed in the network, that achieve the lowest cost. The solution method allows the consideration of several LSP attributes: degree of survivability (link disjoint and node disjoint cases), maximum hop count, usable colours and preferred routes. Computational results show that the procedure can find solutions in low computing times.

References

1. Awduche et al, □A Framework for Internet Traffic Engineering□, *Internet Draft - Work in Progress*, July 2000

2. J. Beasley, “Lagrangean Relaxation”, *Modern Heuristic Techniques for Combinatorial Problem*, Ed. by Colin Reeves, Blacwell Scientific publications, 1993
3. R. Bhandary, “Optimal Diverse Routing in Telecommunication Fibre Networks”, IEEE, 1994
4. S. Choudhury, L. Berry, “Two methods for Minimum Cost Synthesis of Link-disjoint, Hop-limited Virtual Paths in ATM Networks”, ITC 16, pp. 613-622, 1999
5. EURESCOM Project, "Routing and Resource Management in DiffServ", Technical Information, Dec 2000, www.eurescom.de
6. M. Held, P. Wolfe, H. Crowder, “Validation of subgradient optimisation”, *Mathematical Programming*, vol. VI, pp. 62-88, 1974
7. Toni Li, “MPLS and the evolving Internet architecture”, IEEE Communications Magazine, December 1999
8. G. Swallow, “MPLS Advantages for Traffic Engineering”, *IEEE Communications Magazine*, December 1999
9. A. Sousa, R. Valadas, L. Cardoso, A. Duarte, “ATM Network Dimensioning for Mixed Symmetrical and Asymmetrical Services with Dynamic Reconfiguration in a Multi-Network Provider Environment”, *Third IFIP Workshop on Traffic Management and Design of ATM Networks*, England, 1999, pp. 5/1-5/15

DSS: A Deterministic and Scalable QoS Provisioning Scheme

Guillaume Urvoy-Keller and Ernst W. Biersack

Institut Eurecom, 2229, route des Crêtes, 06904 Sophia-Antipolis, France
{urvoy,erbi}@eurecom.fr

Abstract. The design of traffic management schemes for multimedia applications is a challenge for the future Internet. The main problem is the high burstiness of the multimedia traffic. Most of the traffic management schemes try to minimize the total bandwidth required to serve multimedia applications. However, these schemes typically suffer from a poor scalability. Rather than minimizing the burstiness of the multimedia sessions, we present an approach that tries to benefit from high variability of multimedia sessions to maximize the bandwidth offered to the best-effort traffic. The proposed scheme, called DSS, is based on shaping of the flows and on the use of the GPS scheduling policy in the routers. We show that DSS is highly scalable and that its bandwidth requirement for multimedia sessions remains reasonable, in particular smaller than the bandwidth requirement of RPPS.

1 Introduction

Provisioning of Quality of Service (QoS) for multimedia applications in the Internet has received a lot of attention during the last decade. A major difficulty is to accommodate sources with QoS requirements as well as best-effort traffic while maintaining a high level of network bandwidth utilization. The key issues in the design of such a service are: (i) the choice of a service policy (see [1] for an extensive review of the service policies) and (ii) the design of the corresponding admission control algorithm.
However, defining these two elements is not enough. Another important parameter is the scalability of the solution. We say: *a service is scalable if the effort to provide this service does not depend on the number of admitted sessions.* Note that the requirement for scalability is not restricted to the data-transfer phase of a communication, but also applies to the admission control phase.
In this paper, we propose a traffic management scheme that guarantees deterministic QoS to multimedia applications in a scalable way. Sources are assumed to be leaky-bucket constrained with a maximal end-to-end delay requirement. We assume a fluid model that closely approximates the behavior of a packet network with a small packet size compared to the service rate of the servers. The fluid model enables us to concentrate on the central issues. However, recent studies [2,3,4] have emphasized the complexity of deriving bounds in a packet network due to the discrete nature of the problem. Since all our results are derived for the case of a fluid flow model, we will discuss this point.
Our service works as follows: (i) sources are shaped adequately prior to their entrance in the network and (ii) each server reserves an amount of bandwidth equal to the sum of

the peak rate of the shaped sources. The simplicity of the scheme helps to assure its scalability. Shaping allows to limit the peak rate of the sources in the network. Guaranteeing the resource at each server is achieved with the GPS scheduling policy. Each server will serve two kinds of sources: sources that use the QoS provisioning service and best-effort sources. Since the sources with QoS constraints are variable bit rate sources, using GPS allows to redistribute dynamically the unused bandwidth to the best effort sources. There lies the key idea of our scheme: rather than to maximize the number of sources that can be served simultaneously by the QoS provisioning scheme, a method that often leads to the loss of scalability, we propose a simple and scalable QoS provisioning scheme that tries to maximize the bandwidth provided to the best-effort sources.
The remainder of this paper is organized as follows. In Section 2, we recall the main features of the GPS scheduling policy. In Section 3, we present our QoS provisioning scheme, called DSS for Deterministic Shaping Scheme. In Section 4, we review the relevant studies on the GPS policy concerning the delay bounds and the admission control procedures and show their lack of scalability. We also compare our approach with the one of Reisslein et al. [5], which uses the same deterministic shaping but relies on a statistical bufferless multiplexing to achieve a high bandwidth utilization. In Section 5, we evaluate DSS. We first compare DSS with RPPS and show that DSS requires less bandwidth than RPPS. We also evaluate the ability of DSS to redistribute the bandwidth unused by the sources requiring QoS to best-effort sources. In Section 6, we conclude and provide some insights for future work.

2 Properties of the GPS Scheduler

Generalized Processor Sharing (GPS) is the general model of a work-conserving scheduling policy where each session is assigned a weight and is served according to its relative weight among the backlogged sources. GPS assumes a fluid model of all flows, which have to be infinitely divisible. Let $S_i(t,\tau)$ denote the amount of work received by session i ($i \in \{1,\ldots,N\}$) during $[t,\tau]$ and Φ_i its weight. Then, if session i is continuously backlogged during $[t,\tau]$, we have:

$$\frac{S_i(t,\tau)}{S_j(t,\tau)} \geq \frac{\Phi_i}{\Phi_j} \forall j \in \{1,\ldots,N\} \tag{1}$$

Summing equation (1) over all active sessions j, we obtain:

$$S_i(t,\tau) \geq \frac{\Phi_i}{\sum_j \Phi_j} C = r_i^{\min} \tag{2}$$

Equation (2) means that each session i is guaranteed a minimum service rate $r_i^{\min}$ independently of the behavior of the other active sessions (Isolation property). Equation (1) further indicates that GPS has the ability to distribute unused bandwidth between the active sessions consistently with their weight assignment. These features, namely isolation between sources and controlled distribution of the server bandwidth, explain why GPS is so attractive in the context of QoS provisioning.
GPS is also popular due to the delay bound it is able to guarantee. The first results concerning the delay bounds achievable with GPS were obtained by Parekh et al. [6].

Bounds are provided for leaky-bucket constrained sources in the particular case of Rate Proportional Processor Sharing (RPPS). RPPS is a special case of GPS where each source S_i is assigned a weight Φ_i equal to its mean rate R_i, i.e. $\Phi_i = R_i$. Assuming a fluid-flow model and neglecting constant propagation delays, the delay bound in [6] for source S_i with leaky bucket parameters R_i (mean rate) and M_i (maximum burst size), may be written as follows:

$$d_i^{\text{eff}} = \frac{M_i}{\min\limits_{j \in \{1,\ldots,K\}} r_i^{\min}(j)} \tag{3}$$

where $r_i^{\min}(j)$ is the minimum service rate guaranteed at node $j \in \{1,\ldots,K\}$ to source S_i. Note that the end-to-end delay in equation (3) is not simply the sum of local bounds. GPS is only a model of procedure, which is not implementable in a packet network since it assumes that each flow is infinitely divisible (fluid-flow model). A scheme, called Packet Generalized Processor Sharing (PGPS) [7,6], has been proposed to emulate GPS in a packet network. The emulation used by PGPS is cumbersome and lots of research has concentrated on reducing the computational complexity of GPS emulations [8,9,10].

3 DSS: Deterministic Shaping Service

3.1 Leaky Bucket Constrained Sources

We consider sources that are leaky bucket constrained with an additional constraint on their peak rate. A traffic descriptor for a given source S comprises three parameters (p, R, M), which are respectively the peak rate, the mean rate and the maximum burst size of the source. Such a source is able to pass through the leaky bucket controller depicted in Figure 1 without experiencing any loss. The size of the token bucket is $M' = M\frac{p-R}{p} < M$ since the peak rate of the source is finite. Our traffic management scheme is based on deterministic QoS guarantees. A given source S has thus one QoS constraint, which is its maximum end-to-end delay requirement (the traffic management scheme ensures a zero loss rate).

3.2 Deterministic Effective Bandwidth

Central to our proposal is the concept of *deterministic effective bandwidth*. It has been first introduced by Le Boudec [11] in the context of the Network Calculus [12]. The Network Calculus provides deterministic bounds on end-to-end delays and backlogs. A source is modeled through an arrival curve that represents an upper bound on the volume of traffic it can send during any time interval τ. In the case of a leaky bucket constrained source S with parameters (p, R, M), an arrival curve α is (see [11]):

$$\alpha(\tau) = \min\left(p\tau, R\tau + M\frac{p-R}{p}\right),\ \tau \geq 0 \tag{4}$$

Servers are modeled through a service curve representing a lower bound on the service they are able to provide during some time intervals. For instance, a service curve β for

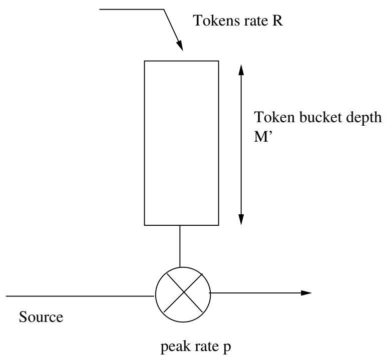


Fig. 1. Leaky bucket controller

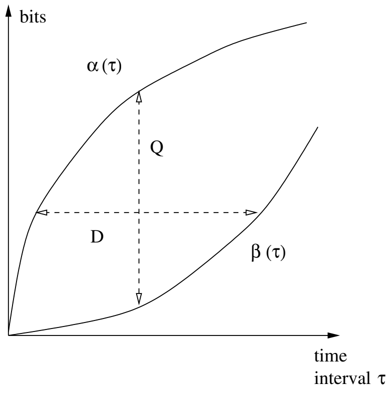


Fig. 2. Upper bound on backlog and virtual delay

a FIFO server (and, more generally, for any server implementing a work-conserving policy) with a service rate C is $\beta(\tau) = C \cdot \tau$ (the time intervals, here, can be the backlog periods). Consider now a system with an input flow characterized by an arrival curve α and a server with a service curve β. An upper bound D on virtual delay (resp. Q on backlog), which corresponds to a real delay if the system works in a FIFO manner, is given by the maximal horizontal (resp. vertical) distance (see Figure 2) between the arrival curve and the service curve of the system (Theorems 1 and 2 of [12]). The concept of deterministic effective bandwidth makes use of these theorems. The deterministic effective bandwidth $e_d(\alpha)$ associated to a given source S for a given arrival curve α and a delay constraint d is the minimum service rate that ensures to this source a delay smaller than d. The following result for $e_d(\alpha)$ is given in [11]:

$$e_d(\alpha) = \sup_{s \geq 0} \left(\frac{\alpha(s)}{s+d} \right) \tag{5}$$

In the special case of a leaky bucket constrained source S with an arrival curve given by equation (4) and a delay requirement d, equation (5) may be re-written as (see Figure 3):

$$e_d(\alpha) = \begin{cases} \frac{M}{d+\frac{M}{p}} & \text{if } 0 \leq d \leq M(\frac{1}{R} - \frac{1}{p}) \\ R & \text{if } d \geq M(\frac{1}{R} - \frac{1}{p}) \end{cases} \tag{6}$$

3.3 QoS Provisioning in DSS

With our traffic management scheme, sources are shaped, prior to their entrance in the network, to a rate equal to their deterministic effective bandwidth. This operation may be done using a spacer [13].

Theorem 1. *The choice of the deterministic effective bandwidth as a shaping rate is optimal, in the sense that any smaller shaping rate could lead to a QoS violation.*

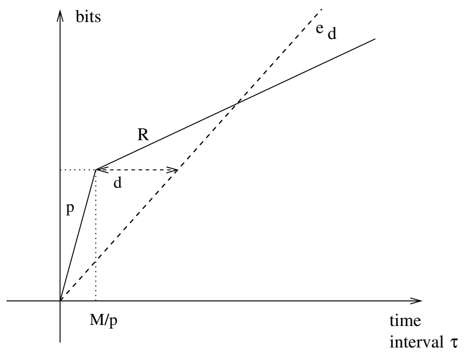


Fig. 3. Deterministic effective bandwidth of a leaky bucket constrained source

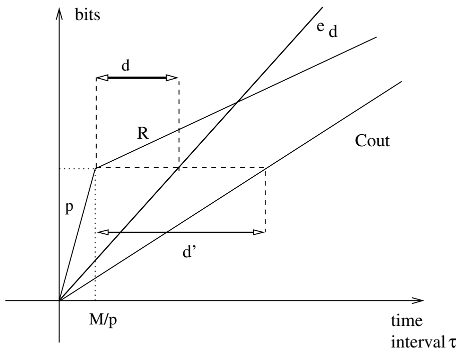


Fig. 4. Choice of the shaping rate

Proof: let us prove the theorem by contradiction. Suppose we choose a shaping rate $C_{out} < e_d(\alpha)$. Let us prove that there is at least one trajectory of the source such that one bit experiences a delay strictly greater than d. Consider the greedy trajectory of S, where S consumes its tokens as soon as they are available. Then, the cumulative arrival rate of this trajectory corresponds to α and the service rate offered by the spacer is $\beta(t) = C_{out} \cdot t$. As a consequence, the bit emitted at time $t = \frac{M}{p}$ experiments a delay d' strictly larger than d (see Figure 4). This proves the result. □

Let us now focus on the behavior of the servers. Our traffic management scheme, DSS, is designed to guarantee a delay equal to zero (within the context of the fluid model) once a source has been accepted in the network. To guarantee a deterministic QoS, we must allocate to each source a service rate equal to its peak rate, i.e. its deterministic effective bandwidth, on each server along its route. This is done by using the GPS policy. For a given server with service rate C, let $(S_i)_{i\in I}$ (resp. $(e_{d_i}(\alpha_i))_{i\in I}$) be the set of sources crossing this server (resp. their deterministic effective bandwidths). The server works as follows: it manages two classes, QoS and BE, with their respective weights Φ_{QoS} and Φ_{BE} defined as follows:

$$\Phi_{QoS} = \sum_{i\in I} e_{d_i}(\alpha_i) \text{ and } \Phi_{BE} = C - \sum_{i\in I} e_{d_i}(\alpha_i)$$

With such a weight assignment, the minimum guaranteed rate $r_{QoS}^{\min}$ for the QoS class is: $r_{QoS}^{\min} = \frac{\Phi_{QoS}}{\Phi_{QoS}+\Phi_{BE}}C = \sum_{i\in I} e_{d_i}(\alpha_i)$. As a consequence, each source using DSS receives, at each server, a minimum service rate equal to its deterministic effective bandwidth. Thus, globally, the DSS network guarantees to each source a minimum service rate equal to its deterministic effective bandwidth. This ensures that the source experiences a delay equal to zero in the network. Moreover, since each server manages only two classes, QoS and BE, (independently of the number of sources actually using DSS) DSS remains scalable during the data-transfer phase. This is noteworthy since the solutions purely based on GPS that will be presented in Section 4 suffer from scalability problems at the data-transfer stage since each server must treat each session individually.

Up to now, we have mainly focused on delay. Another important problem is the buffer space requirement. Since the resource allocation is peak rate based, no buffer space needs to be allocated at the nodes inside the network. This is obviously true only in the context of a strict fluid flow model. Nevertheless, this is a major improvement as compared to other traffic management schemes, like the ones using GPS which, even in a fluid framework, must allocate for each source and at each server a buffer space equal to the maximum burst size of the source (see [1]).

3.4 Admission Control Algorithm

Admitting a New Session: for a given server k with a service rate C_k, let $C_k^{\mathrm{QoS}} = \sum_{i \in I} e_{d_i}(\alpha_i)$ be the sum of the deterministic effective bandwidths of all the sessions in the QoS class (at this server). Consider the admission process of a new source S with a deterministic effective bandwidth $e_d(\alpha)$. The admission algorithm consists in checking the following condition for every node k along the route of source S: $C_k^{\mathrm{QoS}} + e_d(\alpha) \leq C_k$. If the session request can be accepted, then every node k only has to increment C_k^{QoS} by $e_d(\alpha)$. Note also that the conditions to be checked involve only the *total* load of the servers and not the number of previously accepted sources. The admission control is therefore scalable.

Session Termination: servers store only the total service rate allocated to the QoS class. Therefore, the admission control algorithm must ensure that when a session ends, each server along the path of this session receives a message indicating the corresponding deterministic effective bandwidth $e_d(\alpha)$ of the terminating session. Each server k must then decrement C_k^{QoS} by $e_d(\alpha)$.

3.5 Fluid versus Packet Models

DSS has been presented so far in the context of a pure fluid flow model. Recent studies [2,3,4] have emphasized the complexity of deriving bounds in a packet network due to the discrete nature of the problem. Problems arise when some flows are mixed in the same class of service. This is notably the case with DiffServ [14,15]. Specifically, it has been shown in [4] that a deterministic bound may be derived only in the case of a low utilization ρ, with ρ inversely proportional to the maximum hop counts of each flow. This result was obtained in the case of leaky bucket constrained sources and relates more generally to the issue of stability of FIFO networks. Chlamtac et al [2] have derived some necessary conditions for the existence of a finite delay bound in a FIFO network with a general topology in the case of peak rate constrained sources. The authors show that if the peak rate of a source satisfies a constraint related to the number of sources that the source meets on its route, then bounds on end-to-end delays and backlogs exist.
With DSS, each source of the QoS class is treated as a constant bit rate source in the network since resources are reserved based on the peak rate of each source (i.e. their deterministic effective bandwidth). GPS is used in the network to redistribute unused bandwidth to the BE class. From the QoS provisioning point of view, we are in the most favorable case: admission control of constant bit rate sources. However, some

caution needs to be taken when deploying DSS in a real network. We may have to limit the utilization of each link, the number of hops in each path or maybe the number of connection that mutually interfere. This is clearly a complex problem left for future study.

4 Related Work

GPS based approach: DSS uses GPS to *redistribute the unused bandwidth* to the best-effort sources. This is not the way GPS is commonly used. Indeed, GPS is mostly used to offer end-to-end delay bounds. The first results concerning the delay bounds achievable with GPS were derived by Parekh et al. (see Section 2). However, these bounds are restricted to the special case of RPPS. In [17], the authors have extended the result on delay bounds to all CRST (Consistent Relative Session Treatment) networks. Broadly speaking, a CRST network is a GPS network where the relative order between sessions does not vary throughout the network. A session has priority over another one at a given node if its associated weight is higher. CRST includes the case of RPPS and DSS that use the same weight for a source throughout the network. In [17], the authors provide closed-form bounds for arbitrary weight assignments that are compliant with the CRST condition. The key idea is to partition the set of sources in F subsets to form what is called a feasible partition, where the effective service rate received by the sources of subsets $i \in \{1, \ldots, F\}$ depends only on the sources in subset $j < i$. As a consequence, the delay bound obtained for each source will depend on the parameters and weight of the sources belonging to subsets with higher priority. While the result is clearly interesting from an analytical point of view, using it in practice is very complicated: if we were to derive an admission procedure from the delay bounds of [17], it would comprise at least the following operations (assume n sources $S_1, \ldots, S_n$ are already accepted and we try to admit S_{n+1}):

1. Determine the subset j to which S_{n+1} may belong (or create a new subset, depending on its parameters and weight).
2. Re-compute the delay bound associated to all sources (using the closed-form bounds) that belong to subsets $j+1, \ldots, F$, since S_{n+1} affects only the sources belonging to these subsets.
3. Check the non-violation of the delay constraint for each of the $n+1$ sources.

Clearly, such an admission control algorithm is not scalable due to the dependence that exists between the new source and all the sources over which it has priority. Note also that even with the simple RPPS policy, the admission control algorithm is not scalable since the new source affects the minimum service rate of every source whose route has at least one server in common with the route of the new source.

Bufferless multiplexing approach: We saw that QoS provisioning schemes purely based on GPS suffer from scalability problems. Still, the isolation property exhibited by GPS is very attractive. Based on this idea, Reisslein et al. [5,16] introduced a new approach to define a traffic management scheme. The key idea is to shape the traffic prior to its entrance in the network. The shaping rate is chosen subject to the following constraint: minimizing the rate of the source in the network, or, equivalently, ensuring that the maximum delay in the shaper is as close as possible to the required maximum delay. A statistical bufferless multiplexing method is then used to maximize the number of admitted flows. The statistical multiplexing is done via an accurate upper bound on the loss rate experienced by the flows. It allows to accept twice as many sources than RPPS. Reisslein et al. introduced this scheme first for the single node network case [5]. However, to extend the bound on loss to the multiple node network case, sessions must be independent at each node. This is clearly an important limitation since this assumption does not hold in practice. Our approach is similar to this one but with an important difference: rather than maximizing the number of admitted flows, we focus on the redistribution of the unused bandwidth to the best-effort sources.

5 Evaluation of DSS

We have presented DSS in Section 3, emphasizing its scalability during admission control phase as well as during the data-transfer phase. We now evaluate its efficiency. First, we present evidence that DSS requires less bandwidth than RPPS in a network context. In a second stage, we provide some elements to evaluate the ability of DSS to redistribute the bandwidth unused by the QoS class to the BE class. This is an important point since the design goal is to keep our traffic management scalable while trying to provide as much bandwidth as possible to the best-effort traffic.

5.1 Comparison with RPPS

Consider a network with K servers and n sources. We do not make any assumption about the network topology or the routes of the sources in the network, except that the routes are loop-free and that the basic stability conditions are met at each node. For our comparison, we make the following assumptions about the sources:

– all the sources $S_i,\ i \in \{1, \ldots, n\}$ have the same leaky bucket parameters (p, R, M).
– the delay requirement d_i of a given source i is such that the deterministic effective bandwidth of source i is greater than its mean rate $R_i = R$. Using equation (6), we obtain that $d_i \in [0, M(\frac{1}{R} - \frac{1}{p})]$.

For a given server j, let n_j be the number of sources transiting this server. Let us also define two sets associated to server j:

– $I_j = \{i_1, i_2, \ldots, i_{n_j}\}$: set of sources transiting via server j.
– $I_j^{\min}$: set of sources of I_j for which server j is the one providing the minimum guaranteed rate in the RPPS network.

Theorem 2. *For any network compliant with the above assumptions, DSS requires less bandwidth than RPPS.*

Proof: we prove that, at each server, RPPS requires an amount of bandwidth strictly greater than the sum of the deterministic effective bandwidth of each source (which is the service rate required by DSS).
Let us first derive the minimum bandwidth C_j^{DSS} required with DSS at each server j. Let e_i be the deterministic effective bandwidth of S_i, for $i \in \{1, \ldots, n\}$. Using equation (6), we obtain: $e_i = \frac{M}{\frac{M}{p}+d_i}, \forall i \in \{1, \ldots, n\}$. Thus,

$$C_j^{\mathrm{DSS}} = \sum_{i \in I_j} \left(\frac{M}{\frac{M}{p} + d_i} \right) \tag{7}$$

Let us now derive the minimum bandwidth C_j^{RPPS} required with RPPS at server j. Consider a given source $S_i, i \in \{1, \ldots, n\}$. Let $j \in \{1, \ldots, K\}$ be the server, with service rate C_j, providing S_i with a minimum guaranteed service rate $r_i^{\min}$ (see Section 2). Since the weight associated to each source is the same ($\Phi_i = R, \forall i \in \{1, \ldots, n\}$), we obtain:

$$r_i^{\min} \overset{(2)}{=} \frac{R}{n_j \cdot R} C_j = \frac{C_j}{n_j} \tag{8}$$

The effective delay d_i^{eff} provided by RPPS to S_i is :

$$d_i^{\mathrm{eff}} \overset{(3)}{=} \frac{M}{r_i^{\min}} \overset{(8)}{=} \frac{M \cdot n_j}{C_j} \tag{9}$$

S_i is to be accepted if its delay requirement is less than the delay provided by RPPS, i.e. if:

$$d_i^{\mathrm{eff}} \leq d_i \tag{10}$$

Obviously, the minimum service rate C_j^{RPPS} is obtained when equation (10) is an equality. The service rate for server j must thus ensure that $d_i^{\mathrm{eff}} = d_i, \forall i \in I_j^{min}$. However, server j must guarantee to all sources from I_j (even if they do not belong to I_j^{min}) a delay smaller than their required delay. Otherwise, the RPPS network can't guarantee to each source of I_j the maximum delay it requires. Thus, the service rate for server j must ensure that $d_i^{\mathrm{eff}} = d_i, \forall i \in I_j$ and we obtain:

$$C_j^{\mathrm{RPPS}} \overset{(9)}{=} \max_{i \in I_j} \left(M \cdot \frac{n_j}{d_i} \right) = M \frac{n_j}{\min_{i \in I_j}(d_i)} \tag{11}$$

We thus obtain:

$$\begin{aligned}
C_j^{\mathrm{RPPS}} &= \underbrace{\frac{M}{\min_{i \in I_j} d_i} + \ldots + \frac{M}{\min_{i \in I_j} d_i}}_{n_j times} \\
&\geq \frac{M}{d_{i_1}} + \frac{M}{d_{i_2}} \cdots \frac{M}{d_{i_{n_j}}} \\
&> \frac{M}{\frac{M}{p} + d_{i_1}} + \frac{M}{\frac{M}{p} + d_{i_2}} \cdots \frac{M}{\frac{M}{p} + d_{i_{n_j}}} \overset{(7)}{=} C_j^{\mathrm{DSS}}
\end{aligned}$$

This proves the theorem. □

To further quantify how much better DSS performs compared to RPPS in terms of bandwidth consumption (in the context of Theorem 2), we now provide a numerical example. We consider a network with $K = 3$ servers and n sources. We make the following assumptions:

– the mean number of sources crossing each server is the same and is equal to $\bar{n} = 10$. We do not make any assumptions about the routes of the sources,
– the leaky bucket parameters of all the sources are the same, namely (p, R, M),
– the delay requirement d_i of source i is drawn randomly in the interval $]0, M(\frac{1}{R} - \frac{1}{p})[$ using a uniform law. The corresponding deterministic effective bandwidth is e_i. Let also D be the maximum possible value of d_i, i.e. $D = M(\frac{1}{R} - \frac{1}{p})$.

As previously, let n_j be the number of sources at server j and I_j the set of indices of these n_j sources. Let also C^{RPPS} (resp. C^{DSS}) be the sum of the minimum service rates at each of the k servers when RPPS (resp. DSS) is in use. Using equation (11), we obtain:

$$C^{\mathrm{RPPS}} = \sum_{j\in\{1,2,3\}} \frac{M \cdot n_j}{\min_{i\in I_j} d_i} \text{ and } C^{\mathrm{DSS}} = \sum_{j\in\{1,2,3\}} \sum_{i\in I_j} e_i$$

We are interested in the expectations $\overline{C^{\mathrm{RPPS}}} \triangleq E(C^{\mathrm{RPPS}})$ and $\overline{C^{\mathrm{DSS}}} \triangleq E(C^{\mathrm{DSS}})$ of C^{RPPS} and C^{DSS}. Due to the independence between the delay requirements and the number of sources, we obtain:

$$\overline{C^{\mathrm{RPPS}}} = M \cdot k \frac{\bar{n}}{\overline{\min_{i\in I_j} d_i}} \text{ and } \overline{C^{\mathrm{DSS}}} = k \cdot \bar{n} \frac{M}{\overline{d_i} + \frac{M}{p}}$$

Since delay requirements are drawn using a uniform law, we obtain: $\overline{d_i} = \frac{D}{2}$, $\overline{\min_{i\in I_j} d_i} = \frac{D}{k+1}$ (the proof is straightforward using the probability distribution function of the minimum of a set of variables). We are now able to compute $\overline{C^{\mathrm{RPPS}}}$ and $\overline{C^{\mathrm{DSS}}}$ for some values of the leaky bucket parameters of the sources. In Table 1, we present such results for different configurations. These results clearly show that the bandwidth requirement of DSS can be significantly smaller than the one of RPPS.

Table 1. Minimum service rates for RPPS and DSS

(p,R,M)	$\overline{C^{\mathrm{RPPS}}}$	$\overline{C^{\mathrm{DSS}}}$
(100,10,10)	1333.33	545.45
(100,1,10)	121.21	59.40
(100,50,1000)	12000.00	2000.00

5.2 Bandwidth Re-distribution

DSS performs a peak rate allocation for the *QoS* class. However, these sources are not, even after shaping, constant bit rate sources. Using GPS allows to perform a peak rate allocation while redistributing the bandwidth unused by the QoS class. Evaluating this redistribution mechanism is not easy since it is difficult to make hypotheses about the BE class. In the following, we carry out one experiment to illustrate how the bandwidth distribution is performed. We use as criteria:

- $E(S)$: the mean service rate of the BE class,
- $Min(S)$: the minimum service rate of the BE class.

We consider a single server with a service rate C. We assume that DSS is used and that the QoS class comprises n sources $(S_i)_{i\in\{1,\dots,n\}}$. The traffic descriptor of S_i is (p_i, R_i, M_i). Let also e_i be the deterministic effective bandwidth of S_i. The stability of the DSS system ensures that $E(S) = C - \sum_{i\in\{1,\dots,n\}} R_i$. Moreover, since the BE class is granted a weight $\Phi_{BE} = C - \sum_{i\in\{1,\dots,n\}} e_i$, it is guaranteed a minimum service rate $Min(S) = \frac{\Phi_{BE}}{\Phi_{BE}+\Phi_{QoS}} C = C - \sum_{i\in\{1,\dots,n\}} e_i$. These two criteria are interesting since they capture two time scales: a short-term one with $Min(S)$ and a long-term one with $E(S)$. However, it would be interesting to characterize more accurately the dynamics of the system and to evaluate this dynamics for a "worst-case" situation. Such a situation can be the one where the worst-case delay is achieved with a GPS server. It is proven in [7] that the worst-case delays are achieved when all the sources are greedy and synchronous. Going back to our server with service rate C and n sources in the QoS class, we are going to evaluate the service rate that is effectively received by the best-effort class. The effective cumulative service provided by the server corresponds to its service curve offered $\beta(t) = C\cdot t$ since GPS is work-conserving (see Section 3.2) and also since the BE class is assumed to consume all the capacity unused by the QoS class. The cumulative bit rate of the traffic issued by the QoS class corresponds to the arrival curve α characterizing the n sources of the QoS class when they are synchronized. The latter is simply the sum of the individual arrival curves of each source given by equation (4). Thus, one obtains:

$$\alpha(t) = \sum_{i\in\{1,\dots,n\}} \min\left(p_i\cdot t, R_i\cdot t + M_i\frac{p_i - R_i}{p_i}\right)$$

$\alpha(t)$ characterizes the maximum service that the QoS class can require during the worst-case situation. Since $\beta(t)$ characterizes the service provided to both classes (BE and QoS), we deduce that the cumulative service effectively received by the BE class is the $\beta(t) - \alpha(t)$. The latter is represented as the shaded area of Figure 5. A change in the slope of $\alpha(t)$ corresponds to a time instant when a bucket of one source gets empty. The corresponding source, which was previously emitting at its peak rate, now emits at its mean rate. This is why the slope of $\alpha(t)$ decreases with time. When the last bucket gets empty, the slope of $\alpha(t)$ is the sum of the mean rates of all sources. Since the basic stability conditions are met, the vertical distance between $\alpha(t)$ and $\beta(t)$, i.e. $\beta(t) - \alpha(t)$ goes to infinity when t increases. To further illustrate the dynamics of DSS, we now provide a numerical example with a single server with service rate C and $n = 40$ sources in the QoS class. For a given source with parameters (p_i, R_i, M_i), we draw the traffic parameters using a uniform law, as follows: $p_i \in [200, 1000]$ (kbits/s), $R_i \in [10, 50]$ (kbits/s) and $M_i \in [10, 50]$ (kbits). These parameters have been chosen so that the total

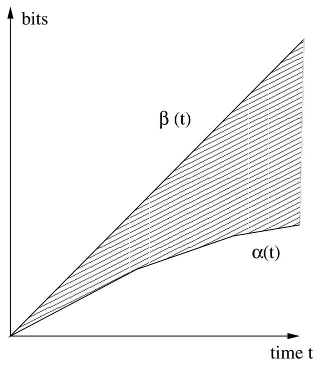


Fig. 5. Cumulative service rate received by the BE class during the "worst-case" period

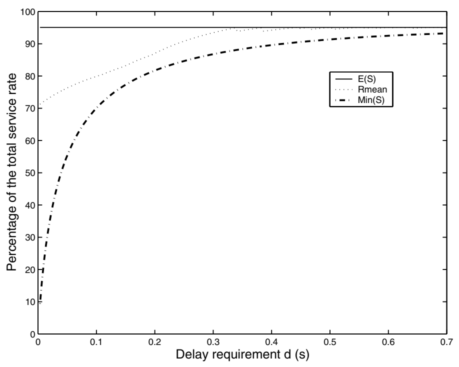


Fig. 6. Mean service rate received during the "worst-case" scenario

mean rate of the QoS class represents a low fraction of the service rate C (about 5% of the service rate) while the total peak rate of the QoS class is close to C. The sources from the QoS class have thus a high variability in terms of bandwidth requirement and we expect that the use of GPS will enable the BE class to receive an important fraction of the unused bandwidth. We assume that the n sources require the same delay bound d and make d vary within an interval $[0, d_{\max}]$. For a given set of sources with parameters $(p_i, R_i, M_i)_{i\in\{1,\ldots,n\}}$, we set: $d_{max} = \max_{i\in\{1,\ldots,n\}}\left(M_i\left(\frac{1}{R_i} - \frac{1}{p_i}\right)\right)$. This choice of $d_{\max}$ ensures that at least one source has a deterministic effective bandwidth strictly greater than its mean rate. Otherwise, if one chooses a value greater than $d_{\max}$, all the sources are shaped at their mean rate and the results are of little interest. We consider the "worst-case" situation presented above and we evaluate the mean percentage R_{mean} of the service rate C received by the BE class during this period. We have to fix a length T for this period. Since all the sources are greedy and synchronous, we choose T as the minimum time it takes to empty the backlogs of all the sources.

The result obtained for a given experiment is presented on Figure 6. All the values are expressed as percentages of the total bandwidth C, which remains unchanged during all the experiments and is chosen to be greater than the sum of all the deterministic effective bandwidths (a requirement of DSS). The upper curve represents the maximum service that may be received by the BE class. This maximum service is equal to $C - \sum_{i\in\{1,\ldots,n\}} R_i$, i.e. $E(S)$, since during the "worst-case" situation, the sources of the QoS class emit at least at their mean rate. The lower curve represents the minimum service rate $Min(S)$ that may be received by the BE class, i.e. $C - \sum_{i\in\{1,\ldots,n\}} e_i$. The latter varies with the delay requirement d since the deterministic effective bandwidths are functions of d. The intermediate curve represents the mean service rate R_{mean} received by the BE class during T. It is interesting to note that even when the delay requirement is low, the mean service received is high, above 70%.

The previous scenario is a realistic one since the burstiness of the sources is high (considering the ratio $\frac{p}{R}$ to characterize the burstiness). Even if it applies only to a single

trajectory, the experiment clearly shows that even during a typical "worst-case" scenario, the percentage of service received by the best-effort class may be high.
From this experiment, we can conclude that there is strong evidence that DSS, by using GPS, efficiently redistributes the bandwidth unused by the QoS class.

6 Conclusion

Recent traffic management schemes that aim at providing QoS to multimedia applications rely on complex mechanisms requiring an individual treatment of each source or based on a complex statistical multiplexing of the sources. The main objective of these traffic management schemes is to maintain the global resource requirement at a reasonable level. However, minimizing the total bandwidth used by the multimedia applications often results in a poor scalability. Things worsen as non-scalability may occur during the data-transfer phase, if each source must be treated individually, or during the admission control phase, if the admission of a new source affects many of the already established sessions.
In order to avoid these scalability problems, we propose a new approach. The key idea is not to minimize the bandwidth consumption of the multimedia sources but rather to redistribute efficiently the bandwidth unused by the multimedia applications. To limit the greediness of the multimedia applications, we first limit as much as possible their peak rate before they enter the network. This is achieved by an adequate shaping of the sources. We then use GPS inside the network, which allows to offer to the multimedia application the bandwidth they need during bursty periods and to redistribute as much bandwidth as possible to the best-effort traffic during periods when the multimedia applications are less active. The resulting traffic management scheme, called DSS, is fully scalable during the data-transfer phase and the admission control phase. We also provide evidence that this scalability is not obtained at the expense of a too important bandwidth consumption: (i) DSS requires less bandwidth than RPPS, the most popular version of GPS and (ii) with DSS, the best-effort traffic is always granted an important amount of bandwidth, even during periods of high activity of the multimedia sessions.
Future work should address a possible formulation of DSS in a statistical setting, which would help to admit more QoS sessions. However, the challenge is to keep the scheme scalable. Making use of a statistical multiplexing seems difficult (see [16]). The notion of statistical effective bandwidth, with a scheme remaining deterministic in the core of network, could be a first step in this direction.

References

1. Zhang, H.: Service discipline for guaranteed performance service in packet-switching networks. Proceedings of the IEEE **83** (1995) 1374 –1396
2. Chlamtac, I., Faragó, A., Zhang, H.: A deterministic approach to the end-to-end analysis of packet flows in connection-oriented networks. IEEE Transactions on Networking (1998) 422–431
3. Charny, A., Le Boudec, J.Y.: Delay bounds in a network with aggregate scheduling. In: Quality of Future Internet Services. Volume 1922 of Lecture Notes in Computer Science., Berlin, Germany, Springer (2000) 1–13

4. Bennett, J., Benson, K., Charny, A., Courtney, W.F., Le Boudec, J.Y.: Delay jitter and packet scale rate guarantee for expedited forwarding. In: Proceedings of IEEE Infocom, Anchorage, Alaska (2001)
5. Reisslein, M., Ross, K., Rajagopal, S.: Guaranteeing statistical QoS to regulated traffic: The single node case. In: Proceedings of IEEE Infocom, New York (1999)
6. Parekh, A.K., Gallager, R.G.: A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. IEEE/ACM Transactions on Networking **2** (1994) 137–150
7. Parekh, A.K., Gallager, R.G.: A generalized processor sharing approach to flow control in integrated services networks: The single node case. IEEE/ACM Transactions on Networking **1** (1993) 344–357
8. Golestani, S.J.: A self-clocked fair queueing scheme for broadband applications. In: Proc. of IEEE INFOCOM'94, Toronto, Canada (1994) 636–646
9. Bennett, J.C.R., Zhang, H.: WF2Q: worst-case fair weighted fair queueing. In: Proceedings of IEEE Infocom, San Fransisco, California (1996) 120–128
10. Shreedhar, M., Varghese, G.: Efficient fair queueing using deficit round robin. ACM Computer Communication Review **25** (1995) 231–242
11. Le Boudec, J.Y: An application of network calculus to guaranteed service networks. IEEE Transactions on Information Theory **44** (1998)
12. Cruz, R.L.: Quality of service guarantees in virtual circuit switched networks. IEEE Journal on Selected Areas in Communications **13** (1995) 1048–1056
13. Boyer, P., Guillemin, F., Servel, M.: The spacer-controller : An efficient upc/npc for atm networks. In: ISS'92. (1992)
14. Blake, S., Black, D., Carlson, M., Davies, E., Z. Wang, Weiss, W.: An architecture for differentiated services. IETF RFC 2475 (1998)
15. Bennett, J., Benson, K., Charny, A., Courtney, W.F., Le Boudec, J.Y., Shiu, A., Kalmanek, C., Stiliadis, D., Fioriru, V., Ramakrishnam, K.K.: An expedited forwarding phb. (Technical Report draft-ietf-diffserv-rfc2598bis-00.txt)
16. Reisslein, M., Ross, K., Rajagopal, S.: Guaranteeing statistical QoS to regulated traffic: The multiple node case. In: 37th IEEE Conference on Decision and Control (CDC), Tampa, Florida (1998)
17. Zhang, Z.L., Liu, Z., Towsley, D.: Closed-form deterministic performance bounds for the generalized processor sharing scheduling discipline. Journal of Combinatorial Optimization, Special Issue on Scheduling **1** (1998)

Experience with an IP QoS Physical Testbed: Problems and Research Issues

Ian F. Akyildiz

Ken Byers Distinguished Chair Professor in Telecommunications
Broadband and Wireless Networking Laboratory
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
ian.akyildiz@ece.gatech.edu

Abstract. Broadband and Wireless Networking Laboratory Next Generation Internet testbed is setup in collaboration with NASA Qbed for prototyping the emerging QoS networking technologies. Internet2's backbone network, Abilene, is used to connect the BWN testbed with several others spread throughout USA. The testbed has been setup to support MPLS Traffic Engineering and Differentiated Services. Experiments have been conducted to gain a better understanding and evaluate the performance of the current protocols to facilitate their deployment in operational networks. We have proposed new schemes for improving end-to-end QoS guarantees and are studying the advantages and disadvantages of using DiffServ and MPLS in a heterogeneous traffic environment. The research will help next generation routers to recognize multimedia traffic types and their QoS expectations not only by research but also experimentally on the physical testbed.

Affordable QoS in Future Wireless Networks: Myth or Reality ?

Jens Zander

Center for Wireless Systems
Royal Institute of Technology, S-100 44 STOCKHOLM, Sweden
Jens.Zander@radio.kth.se

Abstract. Future wireless access system will have features and requirements that are quite distinct form current systems, mostly designed for telephony. Such features include higher bandwidth, mixed, packet oriented and strongly asymmetric traffic patterns as well inhomogeneous network architectures. The question whether we are able to provide meaningful Quality-of Service guarantees in the same way as it is done in wire-line networks at an affordable price remains a open question. In this paper we give an overview over some of the key problems and some the possible solutions for future wireless multimedia systems

1 Introduction

As new wireless systems evolve to complement and replace current 2nd generation wireless access systems (e.g. GSM), a distinct shift in design criteria can be noted. From the being primarily systems for voice communication, future wireless systems will deal with data and will in particular be tailored to different multimedia applications. Due to the large impact of the "web" and web-browser as the common software platform for various IT-applications, provisioning Internet services has become the main design paradigm in defining 3rd and subsequent generations of wireless access systems. Third generation wide-area access systems (e.g. UMTS, FPLMTS) currently in the standardization process and are likely to be deployed commencing in 2002. Such systems are technically capable to provide packet and circuit switched (low level) *bearer services* with on-air data rates between 384 kbit/s (wide area coverage) to 2 Mbit/s(indoor/microcell coverage)[1],[2]. At the same time we can see the evolution of Wireless LAN:s, which now target to provide similar services but at much higher data rates (e.g. HIPERLAN II, 10 Mbit/s and more) in office environments. The question what kind of Quality of Service guarantees can be given in a wireless systems is addressed in this paper. It is shown that classical teletraffic tools such as overproviding resources and using trunking tend to be of limited value in high-speed wireless access system when this aim to provide affordable services anywhere, anytime. Further, we will investigate some of the new distinctive features of future wireless access systems to see what impact these have on the resource management and planning strategies for future wireless multimedia systems. Alternative concepts and architectures for future wireless systems, which can address the above nee ds, will be sketched.

2 Quality-of-Service in Wireless Access systems

The telecommunication industry has a long-standing tradition of defining communication services in terms of Quality-of-Service (QoS) parameters. Typical quantities of interest include statistical measures as bit error probabilities, message delays, throughput and the probability of being denied a service (e.g. blocking probability). In many cases a commercial service is provided with contractual guarantees expressed in one or several of the parameters above. In order fulfill these guarantees, when the demand/traffic load is fluctuating, the operator uses overprovisioning of resources. This means that he will provide more capacity (or more signal power) than actually required in order to handle statistical fluctuations in the demand or in the network performance. Efficient resource management utilizes trunking, i.e. the fact that cleverly aggregate traffic and user demands exhibit less fluctuations. This, in turn, allows for a lower degree of overprovisioning and a better utilization of the communication resources.

With the rapid advances of optical fiber communication techniques, overprovisioning is less of a problem in many wire-line systems. In wireless systems, as we will see, this remains an important problem. Although the technical capabilities of these systems to provide these data rates stands without question, the actual provided data rates may be much lower. This due to the simple fact that the number of access points ("base stations") and thus the cost grows almost linearity with the provided data rate if we want to provide a uniform, "seamless" coverage[3]. In fact it can be shown that the cost follows the expression

$$C_{system} \approx cN_{AP} \approx cN_{user}B_{user}A_{service}f(Q) \quad (1)$$

where N_{AP} is the number of access points ("base stations"), N_{user} is the number of users, B the average data rate of the users, A is the service area covered (or possibly the "volume" indoors). f(Q) is a function of the required quality of service and can be seen as the "overprovisioning margin" required to fulfill the service guarantees. The cost factors generally depend rather weakly on the basic radio technology (e.g. the air interface) employed. This is mainly due to the fact that state-of-the-art modulation & signal processing technology are quite advanced and already so close to the Shannon limits that a radical improvement in processing capabilities will not significantly improve the performance. Clearly, mobile telephony users have gotten used to large coverage areas with high quality of coverage & service availability (anytime, anywhere service), which has been feasible since the bandwidth B has been low.

Keeping $A_{serviced}$, N_{user} and f(Q) constant, it is clear that the cost is directly proportional to the user data rate, or equivalently, the cost per transmitted bit remains the same. As was noted in the introduction, future systems are expected to provide much higher data rates than current systems. Data rates in personal communication systems are certainly limited by propagation conditions as multipath etc., but the primary constraining factor is the link budget, i.e. terminal power consumption[3]. As one could phrase it: "A picture does not only say more than a thousand words" - it is usually a thousand times more power consuming to transmit that picture. Since the required transmitter power increases linearly with the bandwidth, high speed radio access will have but a very limited range. The latter clearly has repercussions on the economics of such systems: either we will have to invest heavily in a dense ubiquitous infrastructure, or be limited to cover only certain areas were we will expect to find users width extensive bandwidth requirements. The conclusion is that straightforward overprovisioning to provide strict QoS guarantees is not an option in multimedia wireless access systems.

What are the possibilities to use efficient resource management to achieve the classic trunking gains and to improve resource utilization? What are the uncertainties we have to fight? Looking at a wireless access system as depicted in fig 1, we can identify the following two main categories of uncertainties:

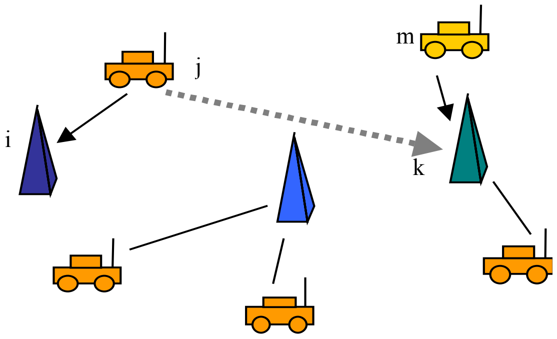


Fig. 1. Wireless Access System

2.1 Traffic/Demand Fluctuations

This includes the classical teletraffic problem of a certain access port ("base station") only having a limited available bandwidth which at times may be exceeded, if too many mobile terminals are accessing it. If the bandwidth is large compared to the individual requirements, e.g. in cellular phone systems, the total instantaneous traffic load on each base station is close to its average value and a very good resource utilization can be achieved. When, however, user bandwidth requirements are high compared to the total bandwidth available in an access port and in addition if the demands strongly vary between users, the story is different. To illustrate this, we take the UMTS/UTRA system as an example where each access port has a total bandwidth limitation of about 2 Mbit/s per carrier. If we would have several users requiring 384 kbits/s or more, it is easy to see that the total number of users that can be served is rather limited and load fluctuation are likely to be significant.

The above problem can be dealt within the classical setting of traffic theory. However, what complicates the situation further, is the coupling between the different access port-mobile terminal links in different "cells" through *interference*. Returning to figure 1, while communicating with access port i, terminal j will also interfere with the reception of terminals m communication with access port k. The level of disturbance will depend on the propagation characteristics of the difference (cross-) links but also on the user requirements of user j. The higher data rate, the higher transmitter power, the higher is the disturbance level at access port k. If the user requirements reflect multimedia traffic characteristics, large and strongly varying data rate requirements, the interference will correspondingly exhibit strong variations and require large received power margins which in turn lead to a poor resource utilization. This is in contradiction with classical cellular phone systems where statistical averaging works excellent also for the interference since it is constructed of many sources with similar power.

2.2 User Location/Propagation Uncertainties

Another complication not dealt with in classical teletraffic theory is the user mobility. In a mobile communication system, QoS guarantees have to be provided without knowing where the users is going to be located while accessing the services. Due to the difficult to predict movements of the user, the propagation conditions may vary significantly and achievable data may even become much lower than the specified maximum rates of the equipment - even in the absence of other users competing for the resources. The movements of other terminals will also effect interference levels and add to the fluctuations described in the previous sections. Signal-to-interference level fluctuations in the order of several orders of magnitude within seconds are not uncommon.

2.3 Non-cooperative Interference

Deregulation on the telecommunication arena is expected to continue. Spectrum allocations without exclusive rights for an operator to use a particular band, so called "unlicensed operation" is becoming a popular method of spectrum management. Examples include Wireless LANs in the 2.4 and 5 GHz bands. Whereas the interference in conventional system may be managed in some sense by centralize control, this is not the case in an unlicensed scenario.

Radio resource management techniques have been studied during the last decade to handle the traffic and in particular the interference variations[5],[4]. The key resource management problems in “multimedia”-type system are related to the data rates and delay constraints traffic in small cell environments will exhibit very large peek to average capacity demands. A key problem in system with several users or user groups with different QoS requirements, is that it is mostly not obvious how to combine these quality criteria into a single performance measure or cost function allowing a straight-forward mathematical optimization formulation.

The key to efficient resource utilization is statistical averaging. If the users require high instantaneous bandwidths the system bandwidth has to be large compared to the user data rates to allow efficient "trunking". Due to spectrum limitations this remains a significant problem. The other problem of fluctuating demand and interference has been address by large scale, centralized resource management. Such resource allocation schemes are not unproblematic due to the large amount of signaling and exchange of measurement information they incur.

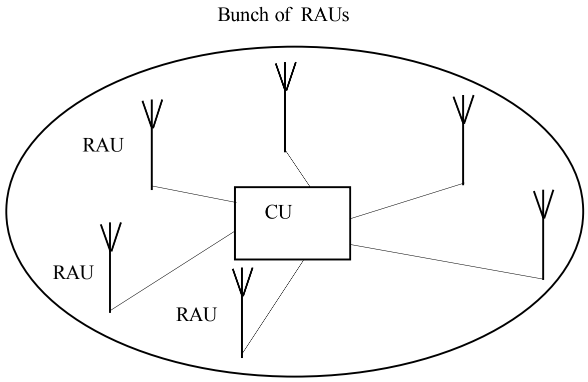


Fig. 2. Bunch consisting of a Central Unit (CU) and a number of Remote Antenna Units (RAU).

An attempt to resolve this problem has been proposed within the ACTS FRAMES project[6],[7]. This hybrid scheme has been coined the "bunch concept". A "bunch" consists of a limited number of Remote Antenna Units (RAUs) that are connected to a functional entity named Central Unit (CU). All intelligence as well as a significant part of the signal processing are located in the CU. The RAUs are simple antenna units capable of transmitting and receiving user signals as well as performing measurements ordered by the CU (Fig. 2). A bunch can cover for instance a group of streets, a building or even a building floor. The relatively short distances between the RAUs and the CU make it feasible to use a high-speed access network. It should also be possible to exchange information between bunches, but on much lower bandwidth. The CU has knowledge about all allocated resources, transmitter powers and path gains in the bunch, and can adaptively allocate resources to the RAUs according to the current need. This results in a very efficient resource utilization within the bunch. One example, considered in the FRAMES project, is to use frequency or time-slot hopping. Initial performance evaluation, reported elsewhere in these proceedings, show very promising results in Manhattan type environments[7]

3 Alternative Infrastructure Solutions

Although efficient resource management can to some extent mitigate the increasing load and interference fluctuations, significant problems remain to be solved when providing a anytime, anywhere, affordable wireless infrastructure. If affordable "multimedia" information exchange is to be possible, i.e. higher data rates at constant or lower cost, either some of the other quality parameters have to be compromised or architectures with radically lower cost factors have to be explored. Alternative concepts and architectures for future wireless systems, which can address the above needs, will be necessary. Several possibilities are open when examining equation (1). One would be to reduce the cost of an access point (the parameter c). Unfortunately, this is not only a matter of waiting for Moore's law to provide us equipment at lower and lower cost. Already today, access points/base station costs are no longer dominated by electronic equipment but by other cost as planning, deployment, cabling, housing etc. This means that we are heading for a situation where even if the equipment would be for free the cost per base station would not be significantly lowered. A solution would be a system concept that allows simple and cheap deployment of infrastructure for very high data rates, e.g. installation by users with very limited knowledge of radio engineering. Such systems have already started to appear in the form of wireless local area networks and systems for ad-hoc short range communication between devices, e.g Bluetooth. These systems operate in unlicensed (i.e. non-exclusive) frequency allocations. Providing telecommunication services even where Quality-of-Service(QoS) guarantees are significantly relaxed. In equation (1) this would correspond to lowering the margin f(Q) or reducing the service area to be covered A. The result would be an "organically" growing, *distributed, heterogeneous, ad-hoc infrastructure*, an "internet in the air", which would constitute a major engineering challenge[9].

A key drawback with these architectures is that they provide little or no guarantees for Quality-of-Service in the traditional sense. An interesting architecture for this is the “Infostation”-concept[8] outlining a sparse infrastructure of “information-kiosks”, close to which very high high data rate communication is feasible. Outside the range of the info-stations only low to moderate data rates are available according to the amount of money the user is willing to spend on the communication. The mission is to deliver as many bits as possible as a user quickly passes through the small cell. This concept may be characterized by "infinite bandwidth - almost nowhere, low bandwidth anywhere, anytime". It is clear that conventional real-time delay limited services will be difficult or very expensive to provide in this environment.

The main research challenge is instead to devise applications that have the "feel" of "anytime, anywhere interactive" but work with strongly varying infrastructure capabilities ("infinite bandwidth almost nowhere - low bandwidth everywhere"). By using so called "memory intensive applications" adaptive software and caching of information, many non-realtime applications can provide the illusion of immediate information retrieval.

Ideas have been presented in regarding so called *context aware* services. One such approach is Smart Media where service knowledge meta-data is added to the multimedia content (creating so-called Smart Media) enabling mobile agents to take decisions about routing, deferring, or altering the mode of operation for on-going communication[10]. A testbed for such services has been established in Stockholm0.

The key feature of such systems are that they exploit the fact that memories are getting larger and cheaper a fast pace, where as a more potent infrastructure is expensive and takes a long time to construct. Putting it bluntly: Moore's law works for electronic devices, not for concrete and antenna towers.

4 Summary & Discussion

Above we have discussed some key features of future wireless multimedia communication and their impact on Quality-of-Service guarantees. We have shown that increasing bandwidths and strongly varying user demands significantly increase the required capacity margins and cause poor resource utilization. Since large bandwidths are expensive in wireless systems, overprovisioning is not an interesting option. Efficient Radio Resource management could mitigate some of the problems, if larger system bandwidths can be utilized and centralized, wide-area RRM schemes are employed. Such solutions are however not problem-free and not well suited for unlicensed systems.

As alternative, truly low cost architectures were discussed. Here the Quality-of-Service requirements are relaxed significantly as well as the concept of uniform QoS over the service area. Instead, deficiencies in the infrastructure capabilities have to be made up for in the applications and terminal. This concept constitutes a significant challenge to traditional telecommunication systems design.

The key question here is: Should the access port infrastructure be very dense (and costly) allowing for "dumb", cheap, low power terminals or should terminals be more complex allowing for the rapid deployment of a cheap infrastructure at the expense of battery life and terminal cost?

References

[1] Ojanperä, T, Prasad, R, "An Overview of Third-Generation Wireless Personal Communications: A European Perspective", *IEEE Personal Comm. Mag.,* Dec 1998.

[2] Prasad, R, Mohr, W, Konhäuser,W, eds., "3rd Generation Mobile Communication Systems", Artech House, 2000.

[3] Zander, J, "On the Cost Structure of Future Wideband Wireless Access", *IEEE VTC '97*, Phoenix, AZ, May 1997.

[4] Katzela, I., Naghishneh, M, "Channel Allocation Schemes for Cellular Mobile Telecommunication Systems: A Comprehensive Survey", *IEEE Personal Communications,* June 1996, pp 10-31.

[5] Zander, J, Kim S-L, "Radio Resource Management in Wireless Networks", *Artech House, 2001.*

[6] Berg, M, Pettersson, S, Zander, J,"A Radio Resource Management concept for "Bunched" Hierarchical Systems", *Proc MMT97,* Melbourne, Dec 1997

[7] Berg, M, "A Concept for Hybrid Random/Dynamic Radio Resource Management", *PIMRC '98*, Boston, USA, Sept. 1998.

[8] Goodman, D, Borras, J., Mandayam, N, Yates, R., "INFOSTATIONS: A New System Model for Data Messaging Services", *IEEE 47ndVeh Tech Conf,* VTC97, Phoenix, AZ, May 1997

[9] M. Flament, F Gessler, F Lagergren, O Queseth, R Stridh, M Unbehaun, J Wu, J Zander: "An Approach to 4th Generation Wireless Infrastructures - Scenarios and Key Research Issues" , *IEEE VTC 99*, Houston, TX, May 1999.

[10] T Kanter, H. Gustafsson , "Active Context Memory for Service Instantiation in Mobile Computing Applications", Proceedings of the Sixth IEEE International Workshop on MobileMultimedia Communications (MoMuC'99). IEEE,ISBN 0-7803-5904-6 (p. 179-183). 1999.

[11] T Kanter , T Rindborg , D Sahlin, "Do-It-Yourself-4G and Enabled Adaptive Mobile Multimedia Communication", Infocom 2001.

Author Index